DATA DEPENDENT RANDOM PROJECTIONS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Tse Lin Keegan Kang

May 2017

DATA DEPENDENT RANDOM PROJECTIONS

Tse Lin Keegan Kang, Ph.D.

Cornell University 2017

Random projections is a technique used primarily in dimension reduction, in order to estimate distances in data. They can be thought of a linear transformation mapping a data matrix X to a lower dimensional space, where distances are preserved in expectation. However, the preservation of distances can be thought of a stepping stone to some eventual goal, such as classification [5, 18], hypothesis testing [15, 24], information retrieval [12, 13], or even reconstructing principal components of data[20, 22]. In this thesis, I will give a background of the basic random projection algorithm. Next, I then look at the structure of random projection matrices and propose modifications to result in a more accurate estimation of distances, which would help in information retrieval and reconstruction of principal components. Finally, I show that it is possible to juxtapose the use of Monte Carlo variance reduction methods with random projections to improve the accuracy of distance estimates, which can then be used in an algorithm or procedure of the users' choice. Theoretical justifications are given, and empirical results are shown with synthetic data, and experiments from publicly available datasets.

BIOGRAPHICAL SKETCH

Keegan Kang is a fifth year PhD student graduating from Cornell University in Spring 2017.

Keegan graduated from Anglo Chinese Junior College in Singapore and did two years of compulsory military service before starting his undergraduate education at the University of Warwick. He spent four years pursuing MMORSE (Masters in Mathematics, Operations Research, Statistics and Economics) and graduated with a First Class Honours in 2012.

During his time at Warwick, he actively participated in student societies like the SSLC (Staff Student Liaison Committee) and the MORSE Society, becoming the Chair of the SSLC (2010-2012) and the President of MORSE Society (2010-2011). He is also heavily involved in mathematics education, having spent one semester doing secondary school teaching mathematics at a local secondary school, and summer in the township of Alexandra teaching mathematics to high school learners and conducting master classes for the teachers under the Warwick in Africa programme. This led to Keegan winning the Giving to Warwick student prize and the Warwick Advantage Gold Award in 2011.

At Cornell University, Keegan has taught a plethora of courses as a graduate teaching assistant, which includes Introductory Statistics, Linear Algebra for Engineers, and Statistical Computing. In 2017, he was awarded the Outstanding Graduate Teaching Assistant award by the Department of Biological Statistics and Computational Biology.

Keegan will be a faculty fellow at the Singapore University of Technology and Design (SUTD) come this summer.

This document is dedicated to all Cornell graduate students. May you all live in interesting times.

ACKNOWLEDGEMENTS

Many thanks go to my advisors Giles Hooker, Karthik Sridharan, and David Mimno who have provided me with many ideas in my research career at Cornell. The students of BTRY 3520 (a course I TA every semester) get a special mention for asking well-thought out questions on Piazza, which in turn open up research avenues I have not considered. A shoutout to the players and imms on 4 Dimensions MUD whom I have had profitable discussions about general theory. Then again, 4 Dimensions MUD is still one of the more unique MUDs out there where the majority of the players are professionals in the tech industry and even sport some PhD holders.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Everything starts somewhere, though many physicists disagree. But people have always been dimly aware of the problem with the start of things. They wonder how the snowplough driver gets to work, or how the makers of dictionaries look up the spelling of words. - Terry Pratchett, Hogfather [21]

1.1 A Brief History On Random Projections

One of the first early works on random projections first appeared as a paper on computer science: *Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality* [8] in an information retrieval context.

Using a result from Johnson and Lindenstrauss [9] (now called the Johnson-Lindenstrauss Lemma, which we will cover in Chapter 2), the authors described a method to solve the approximate nearest neighbor problem with tight bounds on the error. This method was considered groundbreaking at the time, as it ran an order of magnitude faster than known approximate nearest neighbor algorithms.

However, the only reference to random projections in this paper comes from the sentence: "*An elegant technique for reducing complexity owing to dimensionality is to project the points into a random subspace of lower dimension, e.g. by projecting P onto a small collection of random lines through the origin.*" and the rest of this paper describes the approximate nearest neighbors algorithm.

Scant mention was made of the generation of such random projections, ex-

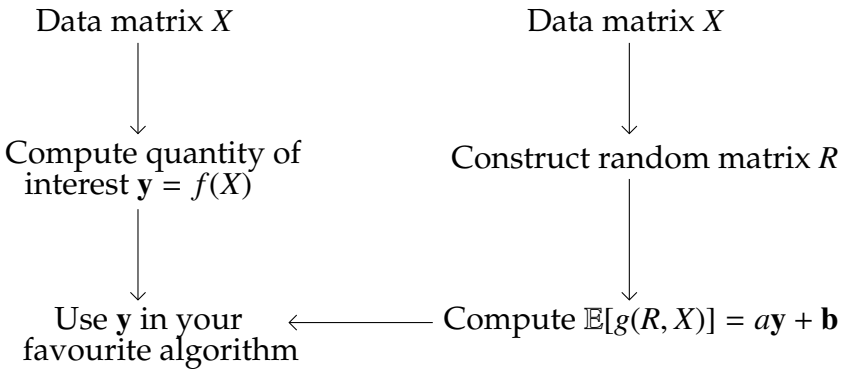

Figure 1.1: Random Projection Flowchart

cept the statement that generating the random lines was equivalent to simulating $\mathbf{v}_i$ vectors from a d dimensional Gaussian distribution $N(\mathbf{0}, I_d)$.

In fact, a casual read of this paper would give the reader the impression that random projections were just a tool that required high technical ability in measure theory to utilize.

It is then rather surprising that in the next few years, there were many papers on random projections showing how they can be used in a wide range of applications appeared, running the gamut from classification to hypothesis testing [5, 12, 13].

We believe that this change happened due to papers by Achlioptas [1] and Dasgupta [5] in the early 2000s, where random projections were portrayed as matrix multiplication - multiplying a constant matrix X by a random matrix R, where the entries r_{ij} are generated from some distribution.

These papers tend to follow a structure similar to Figure 1.1.

First, an algorithm would be identified where some quantity $\mathbf{y}$ is computed

from the data matrix X. For example, $\mathbf{y}$ could be the inner product in the kernel trick for SVMs, or even Euclidean distances between any pairs of points for clustering algorithms.

Next, a random matrix R would be constructed, and an *ansatz* $g(R, X)$ would be hypothesized. The goal would be to show that the expectation $\mathbb{E}[g(R, X)]$ would evaluate to some $a\mathbf{y} + \mathbf{b}$, where a is some scaling factor, and $\mathbf{b}$ some bias.

From a statistical viewpoint, the random matrix R in Figure 1.1 opens the door to many questions, such as the following:

1. How are these entries r_{ij} generated?
2. Are r_{ij} independent or correlated?
3. What distribution do they come from?
4. Is there a fast way to generate these variables?

Furthermore, by looking at the expression $\mathbb{E}[g(R, X)]$, there are even more questions:

1. What is the final distribution of $g(R, X)$?
2. Can statistical theory tell us something strong about $\mathbb{E}[g(R, X)]$?
3. How about the scaling factor a, and the bias $\mathbf{b}$?
4. Are there statistical techniques to improve the random projection estimates of $\mathbb{E}[g(R, X)]$?

The answers to these questions have been answered over the last decade. In this thesis, we will make these questions more precise, come up with more statistical oriented questions, and answer them as well.

```
for each survey respondent do
    Generate a Bernoulli random variable B with p = 1/2
    if B = 1 then
        answer truthfully
    else
        always pick the first choice, e.g. favorite color red
    end
end
```

Figure 1.2: Binary Survey Algorithm

1.2 The Concept Of Randomness

At a first glance, it may be counter-intuitive on how something that *sounds* random can be applied to so many fields with great success. One way to think about this is that random projections are used to *estimate some quantity of interest via expectations*, which will be elaborated on in future chapters.

This idea is not new, and has existed even before random projections in survey analysis, which can be seen in a 1960 paper [26], pre-dating random projections by a few decades. We briefly describe an application of this example.

Example 1.2.1. *A non-random projection example*
Controversial questions in surveys may not be answered truthfully, even if the survey is guaranteed to be anonymous. For example, respondents may be reluctant to state a political party (or candidate) they support even if anonymity is guaranteed, if there are fears of hacks or leaks during extremely polarized elections.

For a binary survey question with only two responses (favorite color red) or (favorite color blue), consider the algorithm in Figure 4.4.

Under the assumption that the survey was conducted appropriately and respondents followed instructions, aggregating the data gives us information about the population's preferences, yet the respondent's answer provides no information about the respondent's political preference.

Here is the mathematics behind this.

Suppose π is the true parameter we want to estimate, and X_i as the response of each person. Denote n_1 the number of people responding `Yes` (denoted as 1), and $n - n_1$ the number of people responding `No` (denoted as 0).

$$\mathbb{P}[X_i = 1] = \frac{1}{2}(\pi + 1 - \pi) + \frac{1}{2}\pi \tag{1.1}$$

$$\mathbb{P}[X_i = 0] = \frac{1}{2}(1 - \pi) \tag{1.2}$$

The log-likelihood of π gives:

$$\hat{\pi} = \frac{2n_1}{n} - 1 \tag{1.3}$$

Taking expectations, the following holds:

$$\mathbb{E}[\hat{\pi}] = \mathbb{E}\left[\frac{2}{n}\sum_{i=1}^{n}\mathbb{E}[X_i] - 1\right] \tag{1.4}$$

$$= \pi \tag{1.5}$$

which gives us an unbiased estimator for π. Furthermore:

$$\text{Var}\,[\hat{\pi}] = \text{Var}\left[\frac{2}{n}n_1\right] \tag{1.6}$$

$$= \frac{4}{n}\text{Var}\,[X_i] \tag{1.7}$$

$$= \frac{4}{n}(\frac{1}{2} + \frac{1}{2}\pi)(\frac{1}{2}(1 - \pi)) \tag{1.8}$$

$$= \frac{1}{n}(1 + \pi)(1 - \pi) \tag{1.9}$$

Therefore, this gives a point estimate of the proportion by computing $\frac{2n_1}{n}$. Confidence intervals can also be constructed around the estimate with the theoretical value of the variance.

In modern days, the above example falls under the field of differential privacy, but is out of the scope of this thesis.

1.3 Thesis Scope And Outline

Taking a second look at Example 1.2.1, we see that there there are no assumptions on *the distribution of people* in the population who hold different viewpoints.

Should the proportion of the population who prefer the color red is a small minority, then it may not be best to generate a Bernoulli random variable $\mathcal{B}$ with $p = \frac{1}{2}$. Rather, the new goal is to generate a Bernoulli random variable $\mathcal{B}$ with $p = \hat{p}$, that reduces our variance of the estimator.

Similarly for random projections, if the distribution of the data X is known, we would like to come up with a random projection matrix R which reduces the error of our estimates.

We will review and formalize the notion of random projections in distance estimation in Chapter 2.

In Chapter 3, we will take a closer look at the random matrix R, and the entries r_{ij}. We will propose changes to the distribution of r_{ij} to better estimate distances based on our data X, and demonstrate this with experiments on syn-

thetic and actual datasets.

Chapter 4 will build upon Chapter 3, changing the structure of the random projection matrix R to reconstruct the principal components and covariance matrix of a dataset X.

We will then cover some tools from Monte Carlo integration, showing how we can improve the estimates of our distances using control variates in Chapter 5.

Finally, we summarize and conclude our results in Chapter 6.

CHAPTER 2

RANDOM PROJECTIONS IN DISTANCE ESTIMATION

In this chapter, we will give an example of how random projections can be used to estimate Euclidean distances and inner products between observations. We then place probabilistic bounds on these estimates, and give some motivation on what the bounds mean. Finally, we give some motivation on why we compute such estimates.

We first define some notation, which will be used throughout the entire thesis.

Let $X_{n\times p} = (x_{ij})$ be a matrix of data collected, where each row $1 \leq i \leq n$ is an observation, and each column $1 \leq j \leq p$ are the covariates.

Let $R_{p\times k} = (r_{ij})$ be a random matrix, where each element r_{ij} are drawn from some distribution, and are not necessarily i.i.d. In this chapter, we will let $r_{ij} \sim N(0, 1)$ to illustrate the basic set up of random projections.

Let $V_{n\times k} = (v_{ij})$ be the matrix formed after multiplying XR.

In most papers involving random projections, the matrix V is usually given by:

$$V = \frac{1}{\sqrt{k}} XR \tag{2.1}$$

We instead use $V := XR$.

In Chapter 1, we stated that we believe the act of writing random projections as matrix multiplication made random projections more accessible to more researchers. Similarly, we feel that writing V without the scaling factor $\frac{1}{\sqrt{k}}$ makes

it easier to use ideas from statistics and Monte Carlo integration on the elements v_{ij}.

2.1 Estimating The Squared Norm Of A Vector

Given a vector $\mathbf{x} := (x_1, x_2, x_3, \ldots, x_p)^T \in \mathbb{R}^p$, the squared norm of $\mathbf{x}$ is given by:

$$\|\mathbf{x}\|_2^2 = \sum_{j=1}^{p} x_j^2 \tag{2.2}$$

Consider a random vector $\mathbf{r} := (r_1, r_2, \ldots, r_p)$, where each r_i is i.i.d. $N(0, 1)$. Set $v := \langle \mathbf{x}, \mathbf{r} \rangle$, the inner product of $\mathbf{x}$ and $\mathbf{r}$. Taking the square of v yields:

$$v^2 = \left(\sum_{j=1}^{p} x_j r_j \right)^2 \tag{2.3}$$

$$= \sum_{j=1}^{p} x_j^2 r_j^2 + 2 \sum_{s=1}^{p-1} \sum_{t=s+1}^{p} x_s r_s x_t r_t \tag{2.4}$$

Using Figure 1.1 (Page 2), as an aid, suppose that:

- The random matrix R corresponds to the random vector $\mathbf{r}$
- The quantity of interest $\mathbf{y}$ corresponds to the norm $\|\mathbf{x}\|_2^2$
- $\mathbb{E}[g(R, X)]$ corresponds to $\mathbb{E}[v^2]$

It is desirable to have $\mathbb{E}[v^2] := a\|\mathbf{x}\|_2^2 + \mathbf{b}$, and indeed:

$$\mathbb{E}[v^2] = \mathbb{E}\left[\sum_{j=1}^{p} x_j^2 r_j^2 + 2 \sum_{s=1}^{p-1} \sum_{t=s+1}^{p} x_s r_s x_t r_t \right] \tag{2.5}$$

$$= \sum_{j=1}^{p} x_j^2 \mathbb{E}[r_j^2] + +2 \sum_{s=1}^{p-1} \sum_{t=s+1}^{p} x_s x_t \mathbb{E}[r_s] \mathbb{E}[r_t] \tag{2.6}$$

$$= \sum_{j=1}^{p} x_j^2 \tag{2.7}$$

with $a = 1, b = 0$.

Now consider the following vector-matrix multiplication:

$$V := \begin{pmatrix} v_1 & \dots & v_k \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & \dots & x_p \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1k} \\ r_{21} & r_{22} & \dots & r_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p1} & r_{p2} & \dots & r_{pk} \end{pmatrix} = XR \tag{2.8}$$

If we square each element of V, then each v_i^2 can be seen as being drawn from some probability distribution V^2 with a mean of $\|\mathbf{x}\|^2$. Using the Law of Large Numbers, computing $\frac{1}{k}\sum_{j=1}^{k} v_j^2$ gives an estimate of $\|\mathbf{x}\|^2$.

Note that if we had used the notation $V := \frac{1}{\sqrt{k}}XR$, then the scaling factor ensures that $\sum_{j=1}^{k} v_j^2$ gives an estimate of the norm we require. However, we then may lose the intuition that statistical theory can give us.

Framed in this context, the second moment of this probability distribution becomes important. If the second moment is low, then fewer draws are needed from this distribution to get a good estimate of $\|\mathbf{x}\|^2$, which implies fewer columns of R.

2.1.1 Probability Bounds In Estimation Of The Norm

The following lemma from Vempala [25] allows us to place bounds on the estimate of $\|\mathbf{x}\|^2$ based on the number of columns k in R. From a probability theory context, we are asking ourselves how many i.i.d. draws v^2 are needed such that the mean of these estimates are within some relative ϵ of the true mean $\|\mathbf{x}\|_2^2$.

Lemma 2.1.1. Suppose we have $R_{p\times k}$ where each entry r_{ij} are i.i.d. from $N(0,1)$. Consider a vector $\mathbf{x} \in \mathbb{R}^p$, and let $\mathbf{v} = \mathbf{x}R$. Without loss of generality, assume $\|\mathbf{x}\|_2^2 = 1$. Then for any $\epsilon > 0$:

$$\mathbb{P}\left[\left\{\frac{v_1^2 + \ldots + v_k^2}{k} - \|\mathbf{x}\|_2^2\right\} \geq \epsilon\|\mathbf{x}\|_2^2\right] < 2\exp\left\{-(\epsilon^2 - \epsilon^3)\frac{k}{4}\right\} \tag{2.9}$$

To prove this, we will need the following lemma, Markov's Inequality:

Lemma 2.1.2. Given a non-negative random variable X, and a constant $a > 0$, then:

$$\mathbb{P}[X \geq a] \leq \frac{\mathbb{E}[X]}{a} \tag{2.10}$$

We can now prove Lemma 2.1.1.

Proof. First, let us rewrite:

$$S_k := v_1^2 + \ldots + v_k^2 \tag{2.11}$$

Then, we can decompose:

$$\mathbb{P}\left[\left\{\frac{S_k}{k} - \|\mathbf{x}\|_2^2\right\} \geq \epsilon\|\mathbf{x}\|_2^2\right] = \mathbb{P}\left[\frac{S_k}{k} \geq (1+\epsilon)\|\mathbf{x}\|_2^2\right] + \mathbb{P}\left[\frac{S_k}{k} \leq (1-\epsilon)\|\mathbf{x}\|_2^2\right] \tag{2.12}$$

We evaluate the first expression of the RHS in Equation 2.12:

$$\mathbb{P}\left[\frac{S_k}{k} \geq (1+\epsilon)\|\mathbf{x}\|_2^2\right] = \mathbb{P}\left[\frac{S_k}{\|\mathbf{x}\|_2^2} \geq (1+\epsilon)k\right] \tag{2.13}$$

$$= \mathbb{P}\left[\exp\left\{\lambda\frac{S_k}{\|\mathbf{x}\|_2^2}\right\} \geq \exp\left\{(1+\epsilon)k\lambda\right\}\right] \tag{2.14}$$

$$\leq \frac{\mathbb{E}\left[\exp\left\{\lambda\frac{S_k}{\|\mathbf{x}\|_2^2}\right\}\right]}{\exp\left\{(1+\epsilon)k\lambda\right\}} \quad \text{by Lemma 2.1.2} \tag{2.15}$$

$$= \frac{\prod_{j=1}^{k}\mathbb{E}\left[\exp\left\{\frac{\lambda v_j^2}{\|\mathbf{x}\|_2^2}\right\}\right]}{\exp\left\{(1+\epsilon)k\lambda\right\}} \quad \text{since } v_i^2 \text{ are independent} \tag{2.16}$$

$$= \left(\frac{\mathbb{E}\left[\frac{\lambda v_1^2}{\|\mathbf{x}\|_2^2}\right]}{\exp\left\{(1+\epsilon)\lambda\right\}}\right)^k \quad \text{and identically distributed} \tag{2.17}$$

Our goal now is to get an expression for the numerator $\mathbb{E}\left[\exp\left\{\frac{\lambda v_1^2}{\|\mathbf{x}\|_2^2}\right\}\right]$.

Recall that each v_i is a linear combination of some r_{ij}, of which are distributed $N(0,1)$. Thus, v_i must also be distributed $N(0,1)$ as well as we assumed $\|\mathbf{x}\|_2^2 = 1$. Therefore, by taking expectations, we have:

$$\mathbb{E}\left[\exp\left\{\frac{\lambda v_1^2}{\|\mathbf{x}\|_2^2}\right\}\right] = \int_{-\infty}^{\infty}\exp\left\{\lambda v^2\right\}\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{v^2}{2}\right\}\,dv \tag{2.18}$$

$$= \int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}\exp\left\{-\frac{v^2(1-2\lambda)}{2}\right\}\,dv \tag{2.19}$$

$$= \frac{1}{\sqrt{1-2\lambda}}\int_{-\infty}^{\infty}\sqrt{\frac{1-2\lambda}{2\pi}}\exp\left\{-\frac{v^2(1-2\lambda)}{2}\right\}\,dv \tag{2.20}$$

$$= \frac{1}{\sqrt{1-2\lambda}} \tag{2.21}$$

where in Line 2.20, the integrand is 1 since this is the probability density function of $N\left(0, \frac{1}{1-2\lambda}\right)$

Substituting the value of Line 2.21 in Line 2.17, we get:

$$\mathbb{P}\left[\frac{S_k}{k} \geq (1+\epsilon)\|\mathbf{x}\|_2^2\right] \leq \left(\frac{\mathbb{E}\left[\frac{\lambda v_1^2}{\|\mathbf{x}\|_2^2}\right]}{\exp\{(1+\epsilon)\lambda\}}\right)^k \tag{2.22}$$

$$= \left(\frac{\exp\{-2(1+\epsilon)\lambda\}}{1-2\lambda}\right)^{k/2} \tag{2.23}$$

$$\leq ((1+\epsilon)\exp\{-\epsilon\})^{k/2} \quad \text{by choosing the maximal value of } \lambda \tag{2.24}$$

$$\leq \exp\left\{-(\epsilon^2-\epsilon^3)\frac{k}{4}\right\} \quad \text{by Taylor expanding } (1+\epsilon) \tag{2.25}$$

By symmetry, the second expression of the RHS in Equation 2.12 evaluates to:

$$\mathbb{P}\left[\frac{S_k}{k} \leq (1-\epsilon)\|\mathbf{x}\|_2^2\right] \leq \left(\frac{\mathbb{E}\left[\frac{-\lambda v_1^2}{\|\mathbf{x}\|_2^2}\right]}{\exp\{-(1+\epsilon)\lambda\}}\right)^k \tag{2.26}$$

and we can repeat the steps from lines 2.18 to 2.25 to get the same upper bound of $\exp\left\{-(\epsilon^2-\epsilon^3)\frac{k}{4}\right\}$.

Therefore, we have that:

$$\mathbb{P}\left[\left\{\frac{v_1^2+\ldots+v_k^2}{k} - \|\mathbf{x}\|_2^2\right\} \geq \epsilon\|\mathbf{x}\|_2^2\right] < 2\exp\left\{-(\epsilon^2-\epsilon^3)\frac{k}{4}\right\} \tag{2.27}$$

and we are done. □

2.1.2 The Johnson Lindenstrauss Lemma

The Johnson Lindenstrauss Lemma [9] is usually cited to show that random projections preserve distances up to some error. We give the lemma here for completion.

Lemma 2.1.3. The Johnson Lindenstrauss Lemma

There exists a linear map $f : \mathbb{R}^d \to \mathbb{R}^k$ such that for all $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$:

$$(1 - \epsilon)\|\mathbf{u} - \mathbf{v}\|_2^2 \leq \|f(\mathbf{u}) - f(\mathbf{v})\|_2^2 \leq (1 + \epsilon)\|\mathbf{u} - \mathbf{v}\|_2^2 \tag{2.28}$$

where $k = \frac{20 \log n}{\epsilon^2}$, with $\epsilon \in (0, \frac{1}{2})$.

Stated another way, the lemma guarantees the existence that such a linear map (random projection map) exists, and gives tight bounds on the error. However, it does not tell us anything on how to choose such a map. Rather, we have to prove that a given linear map satisfies the conditions, such as we did for Lemma 2.1.1.

2.1.3 Why Compute A Norm Estimate?

The work in the last two subsections culminated in showing that the estimates of the norm of a vector has tight probability bounds.

While we stated and (re)-proved Lemma 2.1.1 from Vempala, we proved this a slightly different way from how the proof was originally presented, by treating the lemma as placing a bound on the sum of random variables. Most of the proofs in this thesis follow a similar flow, where we write a certain expression as the sum of random variables, and then apply Markov's inequality and Taylor expand.

However, the casual reader might have a question at this point. The proofs involved may appear to be redundant as the process of matrix multiplication and computing the estimate of the norm has a time complexity of $O(pk)$. Fur-

thermore, the estimate given despite tight probability bounds is not exact. On the other hand, directly computing the norm has a cost $O(p)$, and is exact.

We will see how this helps us when computing an estimate of the Euclidean distance and the inner product between two vectors.

2.2 Estimating Pairwise Euclidean Distances

Consider the task in computing pairwise Euclidean distances between observations $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^p$ of X.

There are $\begin{pmatrix} n \\ 2 \end{pmatrix} = \frac{n(n-1)}{2}$ such pairs to look at, thus the total time to compute these Euclidean distances are $O(n^2 p)$.

Consider the following matrix-matrix multiplication:

$$V := \begin{pmatrix} v_{11} & \ldots & v_{1k} \\ v_{21} & \ldots & v_{2k} \\ \vdots & \ddots & \vdots \\ v_{n1} & \ldots & v_{nk} \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1p} \\ x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{np} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \ldots & r_{1k} \\ r_{21} & r_{22} & \ldots & r_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p1} & r_{p2} & \ldots & r_{pk} \end{pmatrix} = XR \quad (2.29)$$

Random projections allow us to reduce the above time complexity to $O(npk + n^2 k)$, with a trade-off of getting an approximation to the Euclidean distance, rather than the actual value. Therefore if $p < k$, a speedup is experienced.

This is done by first computing the matrix product XR (which is of time $O(npk)$), and then evaluating:

$$\frac{\sum_{s=1}^{k} (v_{is} - v_{js})^2}{k} \quad (2.30)$$

to get an estimate of the squared Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. Doing this for all pairwise Euclidean distances takes $O(n^2k)$.

Squaring the result gives an estimate of the Euclidean distance.

2.2.1 Probability bounds of Euclidean distances

Lemma 2.2.1. Suppose we have $R_{p\times k}$ where each entry r_{ij} are i.i.d. from $N(0,1)$. Consider vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^p$, and let $\mathbf{v}_1 = \mathbf{x}_1 R$, $\mathbf{v}_2 = \mathbf{x}_2 R$. Then for any $\epsilon > 0$:

$$\mathbb{P}\left[\left\{\frac{\sum_{s=1}^{k}(v_{1s} - v_{2s})^2}{k} - \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2\right\} \geq \epsilon\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2\right] < 2\exp\left\{-(\epsilon^2 - \epsilon^3)\frac{k}{4}\right\} \tag{2.31}$$

Proof. Using Lemma 2.1.1, we necessarily need to set $\mathbf{x} = \mathbf{x}_1 - \mathbf{x}_2$, and we are done. □

2.3 Estimating Pairwise Inner Products

Given the same matrix-matrix product as in Equation 2.2, estimating the inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ for any two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ would mean computing:

$$\frac{\sum_{s=1}^{k}\langle v_{is}, v_{js}\rangle}{k} \tag{2.32}$$

However, computing the probability bounds for the estimates of the inner product requires a different strategy than computing the probability bounds for the estimates of the Euclidean distance.

Lemma 2.3.1. Suppose we have $R_{p\times k}$ where each entry r_{ij} are i.i.d. from $N(0,1)$.

Consider vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^p$, and let $\mathbf{v}_1 = \mathbf{x}_1 R$, $\mathbf{v}_2 = \mathbf{x}_2 R$. Then for any $\epsilon > 0$:

$$\mathbb{P}\left[\left\{\frac{\sum_{s=1}^{k}\langle v_{1s}, v_{2s}\rangle}{k} - \langle \mathbf{x}_1, \mathbf{x}_2\rangle\right\} \geq \epsilon\langle \mathbf{x}_1, \mathbf{x}_2\rangle\right] < 4\exp\left\{-(\epsilon^2 - \epsilon^3)\frac{k}{4}\right\} \tag{2.33}$$

Proof. Using the results of Lemma 2.2.1, we have that

$$\mathbb{P}\left[\left\{\frac{\sum_{s=1}^{k}(v_{1s} - v_{2s})^2}{k} - \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2\right\} \geq \epsilon\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2\right] < 2\exp\left\{-(\epsilon^2 - \epsilon^3)\frac{k}{4}\right\} \tag{2.34}$$

Decomposing the results of Lemma 2.2.1 analogous to how we proved Lemma 2.1.1, we have that:

$$\mathbb{P}\left[\frac{S_k}{k} \geq (1+\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2^2\right] \leq \exp\left\{-(\epsilon^2 - \epsilon^3)\frac{k}{4}\right\} \tag{2.35}$$

$$\mathbb{P}\left[\frac{S_k}{k} \leq (1-\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2^2\right] \leq \exp\left\{-(\epsilon^2 - \epsilon^3)\frac{k}{4}\right\} \tag{2.36}$$

where:

$$S_k = \sum_{s=1}^{k}\left(v_{is} - v_{js}\right)^2 \tag{2.37}$$

Then:

$$\mathbb{P}\left[\frac{S_k}{k} \geq (1+\epsilon)\|\mathbf{x}_i + \mathbf{x}_j\|_2^2\right] \leq \exp\left\{-(\epsilon^2 - \epsilon^3)\frac{k}{4}\right\} \tag{2.38}$$

$$\mathbb{P}\left[\frac{S_k}{k} \leq (1-\epsilon)\|\mathbf{x}_i + \mathbf{x}_j\|_2^2\right] \leq \exp\left\{-(\epsilon^2 - \epsilon^3)\frac{k}{4}\right\} \tag{2.39}$$

must also hold as well.

The probability of any of these four events (2.35, 2.36, 2.38, 2.39) happening must (by the union bound) be bounded by $4\exp\left\{-(\epsilon^2 - \epsilon^3)\frac{k}{4}\right\}$.

Now, using the fact that:

$$\|\mathbf{v}_i - \mathbf{v}_j\|_2^2 = \|\mathbf{v}_i\|_2^2 + \|\mathbf{v}_i\|_2^2 - 2\langle \mathbf{v}_i, \mathbf{v}_j\rangle \tag{2.40}$$

$$\|\mathbf{v}_i + \mathbf{v}_j\|_2^2 = \|\mathbf{v}_j\|_2^2 + \|\mathbf{v}_j\|_2^2 + 2\langle \mathbf{v}_i, \mathbf{v}_j\rangle \tag{2.41}$$

with probability $2\exp\left\{-(\epsilon^2-\epsilon^3)\frac{k}{4}\right\}$ we have:

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle = \frac{1}{4}\left(\|\mathbf{v}_i + \mathbf{v}_j\|_2^2 - \|\mathbf{v}_i - \mathbf{v}_j\|_2^2\right) \tag{2.42}$$

$$\geq \frac{1}{4}\left((1+\epsilon)\|\mathbf{x}_i + \mathbf{x}_j\|_2^2 - (1-\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2^2\right) \tag{2.43}$$

$$= \frac{1}{4}\left(4\langle \mathbf{x}_i, \mathbf{x}_j \rangle - 4\epsilon\langle \mathbf{x}_i, \mathbf{x}_j \rangle\right) \tag{2.44}$$

$$= (1-\epsilon)\langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{2.45}$$

and again with probability $2\exp\left\{-(\epsilon^2-\epsilon^3)\frac{k}{4}\right\}$ we have:

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle = \frac{1}{4}\left(\|\mathbf{v}_i - \mathbf{v}_j\|_2^2 - \|\mathbf{v}_i + \mathbf{v}_j\|_2^2\right) \tag{2.46}$$

$$\leq \frac{1}{4}\left((1-\epsilon)\|\mathbf{x}_i + \mathbf{x}_j\|_2^2 - (1+\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2^2\right) \tag{2.47}$$

$$= \frac{1}{4}\left(4\langle \mathbf{x}_i, \mathbf{x}_j \rangle + 4\epsilon\langle \mathbf{x}_i, \mathbf{x}_j \rangle\right) \tag{2.48}$$

$$= (1+\epsilon)\langle \mathbf{x}_i, \mathbf{x}_j \rangle \tag{2.49}$$

Using the union bound of these two events give us:

$$\mathbb{P}\left[\left\{\frac{\sum_{s=1}^{k}\langle v_{1s}, v_{2s}\rangle}{k} - \langle \mathbf{x}_1, \mathbf{x}_2 \rangle\right\} \geq \epsilon\langle \mathbf{x}_1, \mathbf{x}_2 \rangle\right] < 4\exp\left\{-(\epsilon^2-\epsilon^3)\frac{k}{4}\right\} \tag{2.50}$$

and we are done. □

2.4 Conclusion

In this chapter, we revised the notion of how to compute estimates of pairwise Euclidean distances and pairwise inner products of vectors. We also showed that the overall time taken reduces from $O(n^2p)$ to $O(npk + n^2k)$ while using random projection matrices.

One direct application of this result would be for clustering purposes, where pairwise distances between every point would necessarily need to be computed. Random projections in this case would provide a speedup.

Furthermore, we covered a proof on how to compute probability bounds on the estimated norm, and showed how a process of using Markov's inequality and then Taylor's expansion allowed us to get tight bounds.

We then highlighted a strategy to compute probability bounds on the estimates of Euclidean distances and inner products, given our knowledge of computing probability bounds on the estimates of the norms. This strategy will be used throughout this thesis where relevant, and we give a high level overview of how this strategy works.

Strategy 2.4.1. Given each random projection matrix involving i.i.d. entries r_{ij}, then the goal is to express our probability bounds in terms of the sum of some i.i.d. variables.

The task is then to find some $f_1(\epsilon, k_1)$, $f_2(\epsilon, k_2)$ where:

$$\mathbb{P}\left[\frac{S_k^{\text{norm}}}{k} \geq (1+\epsilon)\|\mathbf{x}\|_2^2\right] \leq f_1(\epsilon, k_1) \tag{2.51}$$

$$\mathbb{P}\left[\frac{S_k^{\text{norm}}}{k} \leq (1-\epsilon)\|\mathbf{x}\|_2^2\right] \leq f_2(\epsilon, k_2) \tag{2.52}$$

Then, following the proof of Lemma 2.2.1 and Lemma 2.3.1, we state probability bounds on the estimates of the norm, Euclidean distance (ED), and inner product (IP):

$$\mathbb{P}\left[(1-\epsilon)\|\mathbf{x}\|_2^2 \leq \frac{S_k^{\text{norm}}}{k} \leq (1+\epsilon)\|\mathbf{x}\|_2^2\right] \leq 1 - f_1(\epsilon, k_1) - f_2(\epsilon, k_2) \tag{2.53}$$

$$\mathbb{P}\left[(1-\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \leq \frac{S_k^{\text{ED}}}{k} \leq (1+\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2^2\right] \leq 1 - f_1(\epsilon, k_1) - f_2(\epsilon, k_2) \tag{2.54}$$

$$\mathbb{P}\left[(1-\epsilon)\langle \mathbf{x}_i, \mathbf{x}_j\rangle \leq \frac{S_k^{\text{IP}}}{k} \leq (1+\epsilon)\langle \mathbf{x}_i, \mathbf{x}_j\rangle_2^2\right] \leq 1 - 2f_1(\epsilon, k_1) - 2f_2(\epsilon, k_2) \tag{2.55}$$

We will now further explore the structure of the random projection matrix in Chapter 3.

CHAPTER 3

STRUCTURE OF RANDOM PROJECTION MATRIX

Most algorithms try to juggle speed, memory use, and accuracy. Given a baseline algorithm with some performance, it is rare to find a second algorithm that is faster, uses less memory, and is more accurate than the baseline. Thus, tradeoffs are generally made.

For the basic random projection algorithm to compute estimates of Euclidean distances and inner products, one can argue that for a slight loss in accuracy, there is an increase in speed ($O(n^2p)$ to $O(npk + n^2k)$), and savings in memory (store $V_{n\times k}$ instead of $X_{n\times p}$).

We can view random projections as an optimization where we want to minimize the error:

$$C = \min \sum_{i=1}^{n} \{f(R, \mathbf{x}_i) - \|\mathbf{x}\|_2^2\} \tag{3.1}$$

or in general:

$$C = \min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \{f(R, \mathbf{x}_i, \mathbf{x}_j) - g(\mathbf{x}_i, \mathbf{x}_j)\} \tag{3.2}$$

Here $g(\mathbf{x}_i, \mathbf{x}_j)$ is something we want to estimate (Euclidean distance, inner product for example), and $f(R, \mathbf{x}_i, \mathbf{x}_j)$ is some function estimating $g(\mathbf{x}_i, \mathbf{x}_j)$ via a linear transformation using a random matrix.

Under a random projection framework, the No Free Lunch theorem [28] tells us that we cannot hope for a special random projection matrix which guarantees us a further improvement in accuracy across all types of datasets. In fact, [17] goes one step further and shows that the performance guarantee for a par-

ticular algorithm is bounded by the proportion of favourable datasets for that algorithm.

Going one step further, we will try to see if changing the structure of the random projection matrix will provide advantages for specific types of datasets. Datasets can be sparse or dense. The first few principal components of a dataset may explain a large proportion of the variance (and other parameters may just be noise), or each principal component contribute little to explaining the proportion of variance.

Knowing something about a dataset then can lead us to pick an appropriate random projection structure.

3.1 Related Work

Much work has been done over the last decade to choose a random projection matrix structure, with the trend towards faster computation. The composition of a dataset (for that matrix structure) is of secondary importance.

In this chapter, we will review how the structure of random projection matrices evolved in the last decade, highlighting important contributions made by researchers. After this review, we will showcase our contributions for the structure of the random projection matrix building upon previous work.

We introduce a new probability distribution in this chapter.

Definition 3.1.1. The Sparse Bernoulli distribution with parameter $s > 0$ is given

by $r \sim SB(s)$ where:

$$r = \begin{cases} \sqrt{s} & \text{with probability } \frac{1}{2s} \\ -\sqrt{s} & \text{with probability } \frac{1}{2s} \\ 0 & \text{with probability } 1 - \frac{1}{s} \end{cases} \tag{3.3}$$

Lemma 3.1.1. The Sparse Bernoulli distribution with $s > 0$ has even moments $\mathbb{E}[r^{2k}] = s^k$, and odd moments $\mathbb{E}[r^{2k+1}] = 0$.

Proof. For any $s > 0$, $k \in \mathbb{N}$, we have odd moments to be:

$$\mathbb{E}[r^{2k+1}] = \frac{1}{2s}(\sqrt{s})^{2k+1} + \frac{1}{2s}(-\sqrt{s})^{2k+1} + 0 \tag{3.4}$$

$$= 0 \tag{3.5}$$

and even moments to be:

$$\mathbb{E}[r^{2k}] = \frac{1}{2s}(\sqrt{s})^{2k} + \frac{1}{2s}(-\sqrt{s})^{2k} + 0 \tag{3.6}$$

$$= \frac{1}{2}s^k + \frac{1}{2}s^k \tag{3.7}$$

$$= s^k \tag{3.8}$$

□

3.1.1 Binary Random Projections

In 2003, Achlioptas [1] came up with the theory of sparse random projections, by drawing i.i.d. elements r_{ij} in R from a Sparse Bernoulli distribution, with parameters $s = 1$ and $s = 3$.

This work was seen as novel since this sped up computational time significantly. As entries in $N(0, 1)$ are dense, each value r_{ij} had to be stored in floating point representation, which made matrix multiplication costly.

On the other hand, with $r_{ij} \sim SB(s)$ where $s = 1, 3$, entries in R could be stored as ± 1 by factoring out $\frac{1}{\sqrt{s}}$. Thus, computing XR where R is a matrix of ± 1 requires just adding and subtracting elements of X. Computing the estimate of the respective Euclidean distances or norms then requires multiplication of a constant scaling factor s.

To see where the scaling factor comes into play, consider Equation 2.2 (Page 15), where we compute the matrix $V = XR$, and have entries r_{ij} to be from SB(s) without any factoring taking place.

Consider any arbitrary $\mathbf{x}, \mathbf{r}$, and their inner product $v := \langle \mathbf{x}, \mathbf{r} \rangle$. Next, compute the square of v and we get:

$$v^2 = \left(\sum_{j=1}^{p} x_j r_j \right)^2 \tag{3.9}$$

$$= \sum_{j=1}^{p} x_j^2 r_j^2 + 2 \sum_{s=1}^{p-1} \sum_{t=s+1}^{p} x_s r_s x_t r_t \tag{3.10}$$

Taking expectations yield:

$$\mathbb{E}[v^2] = \mathbb{E}\left[\sum_{j=1}^{p} x_j^2 r_j^2 + 2 \sum_{s=1}^{p-1} \sum_{t=s+1}^{p} x_s r_s x_t r_t \right] \tag{3.11}$$

$$= \sum_{j=1}^{p} x_j^2 \mathbb{E}[r_j^2] + +2 \sum_{s=1}^{p-1} \sum_{t=s+1}^{p} x_s x_t \mathbb{E}[r_s] \mathbb{E}[r_t] \tag{3.12}$$

$$= s \sum_{j=1}^{p} x_j^2 \tag{3.13}$$

since by Lemma 3.1.1 we have that $\mathbb{E}[r^2] = s$.

Thus, replacing R with entries in ± 1 suffices by multiplying our eventual estimate with s.

Achlioptas proved that if r_{ij} was drawn from $SB(1)$ or $SB(3)$, then the proba-

bility bounds are exactly the same as when r_{ij} is drawn from the Normal distribution.

Lemma 3.1.2. Suppose we have $R_{p\times k}$ where each entry r_{ij} are i.i.d. from $SB(1)$ or i.i.d. from $SB(3)$. Consider a vector $\mathbf{x} \in \mathbb{R}^p$, and let $\mathbf{v} = \mathbf{x}R$. Then for any $\epsilon > 0$:

$$\mathbb{P}\left[\left\{\frac{v_1^2 + \ldots + v_k^2}{k} - \|\mathbf{x}\|_2^2\right\} \geq \epsilon\|\mathbf{x}\|_2^2\right] < 2\exp\left\{-(\epsilon^2 - \epsilon^3)\frac{k}{4}\right\} \tag{3.14}$$

We defer the proof of this until Section 3.1.2, where we prove probability bounds for a general $SB(k)$.

However, there are vectors $\mathbf{x}$ (equivalently $\mathbf{x}_1 - \mathbf{x}_2$) where random projections using $SB(s)$, $s = \{1, 3\}$ which give "more" error, in the sense that the probability of our estimated norm (equivalently Euclidean distance) is further away from the true value of $\|\mathbf{x}\|_2^2$ ($\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2$).

Achlioptas showed that these are vectors of the form $(\alpha, \alpha, \ldots, \alpha)$, where every entry of the vector is the same.

We give an alternative proof than Achlioptas by conditioning on the variance of our estimate $\mathbf{v}^2$, and then taking derivatives to find such bad $\mathbf{x}$.

Proof. To compute the variance $\mathrm{Var}[v^2]$, we require $\mathbb{E}[v^2]$ and $\mathbb{E}[v^4]$. As we already have $\mathbb{E}[v^2]$ from Line 3.13, we will compute $\mathbb{E}[v^4]$. Therefore:

$$\mathbb{E}[v^4] = \mathbb{E}\left[\left(\sum_{j=1}^{p} x_j r_j\right)^4\right] \tag{3.15}$$

$$= s^2 \sum_{i=1}^{p} x_i^4 + 6s^2 \sum_{u=1}^{p-1} \sum_{v=u+1}^{p} x_u^2 x_v^2 \tag{3.16}$$

by directly applying Lemma B.0.1 (from the Appendix) and using the fact that the second moments and fourth moments of the Sparse Bernoulli distribution are s and s^2 from Lemma 3.1.1.

Thus we have:

$$\text{Var}[v^2] = \mathbb{E}[v^4] - \left(\mathbb{E}[v^2]\right)^2 \tag{3.17}$$

$$= s^2\left(\sum_{i=1}^{p} x_i^4 + 6\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} x_u^2 x_v^2\right) - s^2\left(\sum_{j=1}^{p} x_j^2\right)^2 \tag{3.18}$$

$$= s^2\left(\sum_{i=1}^{p} x_i^4 + 6\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} x_u^2 x_v^2\right) - s^2\left(\sum_{j=1}^{p} x_j^4 - 2\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} x_u^2 x_v^2\right) \tag{3.19}$$

$$= 4s^2 \sum_{u=1}^{p-1}\sum_{v=u+1}^{p} x_u^2 x_v^2 \tag{3.20}$$

To maximize this variance, WLOG let $\sum_{i=1}^{p} x_i^2 = 1$. Consider the Lagrangian:

$$g(\mathbf{x}) = \sum_{u=1}^{p-1}\sum_{v=u+1}^{p} x_u^2 x_v^2 - \lambda(\sum_{i=1}^{p} x_i^2 - 1) \tag{3.21}$$

and the partial derivatives of x_i:

$$\frac{dg(\mathbf{x})}{dx_i} = 2x_i \sum_{j\neq i} x_j^2 - 2\lambda x_i \tag{3.22}$$

Setting these partial derivatives to zero gives:

$$\sum_{j\neq i} x_j^2 = \lambda \tag{3.23}$$

which implies all x_j have to be equal by symmetry. □

Thus, we can see that while Achlioptas's binary random projections does give a boost in computational time based on how the random matrix is stored, there are nevertheless some types of vectors $\mathbf{x}$ on which this algorithm performs poorly on.

3.1.2 Very Sparse Random Projections

In 2006, Li [12] noted that drawing r_{ij} with parameter s from the Sparse Bernoulli distribution was equivalent to sampling $\frac{1}{s}$ of the data in X. Li proposed sam-

pling at a more aggressive rate than just $\frac{1}{3}$ of the data when $s = 3$, such as $\frac{1}{\sqrt{p}}$, based on the distribution of the data. For example, distributions with exponential error tail bounds admit a possible $s = \frac{p}{\log p}$ due to the majority of observations being around the center.

In this case, it is not easy to get explicit probability bounds of Equation 3.1.2. Theorem 5.3 of Vempala [25] analyzes and computes these probability bounds when s varies.

Theorem 3.1.1. Suppose we have $R_{p\times k}$ where each entry r_{ij} is i.i.d. from a Sparse Bernoulli distribution with parameter s. Let $\mathbf{x} \in \mathbb{R}^p$, and consider $\mathbf{v} = \mathbf{x}R$.

1. Suppose $\exists\, B > 0$ such that $\mathbb{E}[r^4] \leq B$. Then for any $\epsilon > 0$:

$$\mathbb{P}\left[\left\{\frac{\sum_{s=1}^{k} v_k^2}{k} - \|\mathbf{x}\|_2^2\right\} \leq (1-\epsilon)\|\mathbf{x}\|_2^2\right] < 2\exp\left\{-\frac{k(\epsilon^2-\epsilon^3)}{2(B+1)}\right\} \tag{3.24}$$

2. Suppose $\exists\, L > 0$ such that for any integer $m > 0$, $\mathbb{E}[r^{2m}] \leq \frac{(2m)!}{2^m m!}L^{2m}$. Then for any $\epsilon > 0$:

$$\mathbb{P}\left[\left\{\frac{\sum_{s=1}^{k} v_k^2}{k} - \|\mathbf{x}\|_2^2\right\} \geq (1+\epsilon)L^2\|\mathbf{x}\|_2^2\right] \leq \exp\left\{-\frac{k(\epsilon^2-\epsilon^3)}{4}\right\} \tag{3.25}$$

We omit the proof of this theorem as it can be found in Vempala [25], and is just algebra.

Intuitively, we can think of very sparse random projections as a speed up from sparse random projections, since there will be more zeroes in the matrix R. However, the variance for each estimate necessarily must increase as s increases, since the fourth moments of r_{ij} increases as well. Furthermore, apart from bad vectors $\mathbf{x}$ where all entries are equal to each other, the parameter s necessarily needs to vary based on the sparseness of the data.

Nevertheless very sparse random projections still has good accuracies for nice X once a good s is found.

3.1.3 Hadamard Transformations And The "Hashing Trick"

Ailon and Chazelle [2] came up with the Fast Johnson-Lindenstrauss Transform (FJLT), and then the Subsampled Randomized Hadamard Transform (SRHT) [4], which would be used for sparse X. Their method resulted in a faster construction of R, by using the properties of the Hadamard matrix, and generation of only $p + k$ random variables, as compared to $p \times k$ random variables.

The eventual setup was to pad X such that the number of columns p was to a power of 2, and then find:

$$V = XDHS \tag{3.26}$$

where D was a $p \times p$ diagonal matrix, with the diagonal elements drawn from a Sparse Bernoulli distribution with parameter 1, H a $p \times p$ Hadamard matrix, which could be constructed recursively, and S , a $p \times k$ matrix, where each column is drawn with replacement from the identity matrix I_p. Matrix multiplication would happen recursively, instead of generating the entire H matrix.

We give an example of their setup with a toy example where we have $\mathbf{x} \in \mathbb{R}^4$, $D_{4\times4}$, $H_{4\times4}$ and $S_{4\times2}$ to show how $\mathbf{v} \in \mathbb{R}^2$ is constructed.

Recall that the Hadamard matrix of size 2^k by 2^k can be constructed recur-

sively by:

$$H_{2^0} = [1] \tag{3.27}$$

$$H_{2^n} = \begin{bmatrix} H_{2^{n-1}} & H_{2^{n-1}} \\ H_{2^{n-1}} & -H_{2^{n-1}} \end{bmatrix} \tag{3.28}$$

With $\mathbf{x} \in \mathbb{R}^4$, we have:

$$H_{4\times4} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \tag{3.29}$$

HS in Equation 3.26 is equivalent to selecting k columns of the Hadamard matrix with replacement. Suppose we selected columns 1 and columns 2, resulting in:

$$HS = \begin{pmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{3.30}$$

Then DHS is equivalent of taking the Hadamard product (element wise multiplication) of:

$$\begin{pmatrix} r_1 & r_1 \\ r_2 & r_2 \\ r_3 & r_3 \\ r_4 & r_4 \end{pmatrix} \circ HS = \begin{pmatrix} r_1 & r_1 \\ r_2 & -r_2 \\ r_3 & r_3 \\ r_4 & -r_4 \end{pmatrix} \tag{3.31}$$

Consider the entries of $\mathbf{v} \in \mathbb{R}^2$. We would have:

$$v_1 = x_1 r_1 + x_2 r_2 + x_3 r_3 + x_4 r_4 \tag{3.32}$$

$$v_2 = x_1 r_1 - x_2 r_2 + x_3 r_3 + x_4 r_4 \tag{3.33}$$

which implies

$$v_1^2 = \sum_{k=1}^{4} x_k^2 + 2 \sum_{s=1}^{3} \sum_{t=s+1}^{4} x_s r_s x_t r_t \tag{3.34}$$

$$v_2^2 = \sum_{k=1}^{4} x_k^2 + 2 \sum_{s=1}^{3} \sum_{t=s+1}^{4} (-1)^{s+t} x_s r_s x_t r_t \tag{3.35}$$

$$\tag{3.36}$$

Thus we have:

$$v_1^2 + v_2^2 = \sum_{k=1}^{4} x_k^2 + 2x_1 x_3 r_1 r_3 + 2x_2 x_4 r_2 r_4 \tag{3.37}$$

There are two ways to see this: First, if we took the norm $\|\mathbf{x}_i\|_2^2$ as our quantity of interest, then instead of having k independent estimations, we have k dependent estimations.

The second way to see this is to consider what might happen if we removed the matrix S, and considered the product $\mathbf{w} = \mathbf{x}DH$ where both $\mathbf{x}, \mathbf{w} \in \mathbb{R}^p$. Treating $R = DH$, we must have:

$$R = \begin{pmatrix} r_1 & r_1 & r_1 & r_1 \\ r_2 & -r_2 & r_2 & -r_2 \\ r_3 & r_3 & -r_3 & -r_3 \\ r_4 & -r_4 & -r_4 & r_4 \end{pmatrix} \tag{3.38}$$

and if we expand each w_i^2, where $w_i \in \mathbf{w}$, we get:

$$\frac{w_1^2 + w_2^2 + w_3^2 + w_4^2}{4} = \|\mathbf{x}\|_2^2 \tag{3.39}$$

Thus, by thinking of W^2 as a discrete probability distribution with support of p points where each point is uniformly distributed, then:

$$\mathbb{E}[W^2] = \frac{1}{p}\sum_{i=1}^{p} w_1^2 \tag{3.40}$$

$$= \|\mathbf{x}\|_2^2 \tag{3.41}$$

In this case, we have k independent estimations from W^2.

We now take a look at the "hashing trick" [27], which is an analogue of the SRHT.

The "hashing trick" works by defining two hash functions

$$h : \{1, 2, \ldots, p\} \mapsto \{1, \ldots, k\} \tag{3.42}$$

$$\xi : \{1, 2, \ldots, p\} \mapsto \{\pm 1\} \tag{3.43}$$

Then, a vector $\mathbf{x} \in \mathbb{R}^p$ is mapped to $\mathbf{v} \in \mathbb{R}^k$, with the following construction:

$$v_i = \sum_{j:h(j)=i} \xi(i) x_i \tag{3.44}$$

Suppose for our toy example, we had:

$$h(1) = 1 \qquad h(2) = 2 \qquad h(3) = 1 \qquad h(4) = 2 \tag{3.45}$$

and re-named the variables $\xi(i)$ as r_i. Then this is equivalent to the matrix R being

$$R = \begin{pmatrix} \xi(1) & 0 \\ 0 & \xi(2) \\ \xi(3) & 0 \\ 0 & \xi(4) \end{pmatrix} \tag{3.46}$$

i.e. we would have

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \end{pmatrix} \begin{pmatrix} r_1 & 0 \\ 0 & r_2 \\ r_3 & 0 \\ 0 & r_4 \end{pmatrix} = \begin{pmatrix} v_{11} & v_{12} \end{pmatrix} \tag{3.47}$$

and we get

$$v_1^2 + v_2^2 = \sum_{k=1}^{4} x_k^2 + 2x_1x_3r_1r_3 + 2x_2x_4r_2r_4 \tag{3.48}$$

which is the same form to the SRHT.

Thus, one can think of the SRHT and the "hashing trick" as two sides of the same coin.

In the SRHT, we think of generating p random variables $r_1, \ldots, r_p$ from $SB(1)$, put them in columns, and take their Hadamard product with a randomly chosen subset of the Hadamard matrix. Eg, if $\rho = (r_1, r_2, \ldots, r_p)^T$, then

$$R = \begin{pmatrix} \rho \,\middle|\, \rho \,\middle|\, \ldots \,\middle|\, \rho \end{pmatrix} \circ \begin{pmatrix} H_1 \,\middle|\, H_2 \,\middle|\, \ldots \,\middle|\, H_k \end{pmatrix} \tag{3.49}$$

where $H_1, \ldots, H_k$ are columns drawn with replacement from the Hadamard matrix.

In the "hashing trick", we similarly generate p random variables $r_1, \ldots, r_p$ from $SB(1)$, but instead perform blocking on them by choosing a random hash function h. In the toy example, we had chosen the hash function which gave us

$$R = \begin{pmatrix} r_1 & 0 \\ 0 & r_2 \\ r_3 & 0 \\ 0 & r_4 \end{pmatrix} \tag{3.50}$$

and both constructions of R (after normalizing constants) give the same results.

3.2 Our Contributions - BCD Random Projections

We will extend the ideas in the papers [2, 27] for *BCD random projections*, with a few differences. We will relax the assumption of having k independent copies of our quantity of interest and also use properties of the data matrix X in order to choose our r_{ij}. We do so by presenting proofs of the variances of the lengths of vectors behind BCD random projections.

We take a second look at the SRHT with a statistical perspective and look at the purpose of the matrix S, with a view to estimate the norm $\|\mathbf{x}_i\|$. Consider

$$XDH = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix} \begin{pmatrix} r_1 & r_1 & r_1 & r_1 \\ r_2 & -r_2 & r_2 & -r_2 \\ r_3 & r_3 & -r_3 & -r_3 \\ r_4 & -r_4 & -r_4 & r_4 \end{pmatrix} \tag{3.51}$$

$$= \begin{pmatrix} v_1 & v_2 & v_3 & v_4 \end{pmatrix} \tag{3.52}$$

If we think of v_j belonging to a distribution V, where each v_j has an equal probability of being drawn, then we have that

$$\mathbb{E}[V^2] = \frac{1}{4}v_1^2 + \frac{1}{4}v_2^2 + \frac{1}{4}v_3^2 + \frac{1}{4}v_4^2 \tag{3.53}$$

$$= \sum_{i=1}^{4} x_i^2 \tag{3.54}$$

Since the mean of V^2 is equivalent to our norm $\|\mathbf{x}\|^2$, then by the law of large numbers, the mean of k independent v_i's converges to the true mean of V^2. Thus, the purpose of S is to pick k observations v_i^2 from the full set $v_1^2, \ldots, v_p^2$. The difference however, is that the support of the distribution V under the SRHT is contained within the support of the distribution V under ordinary random

projections. The same idea can be applied to the "hashing trick" as well, but the distribution V under the "hashing trick" has a different support than the SRHT.

However, a problem arises if X is not sufficiently sparse, or if any of the columns x_j are highly skewed, since we would need more observations k in order for the mean to converge to $\|\mathbf{x}\|^2$. Intuitively, if each v_i^2 are "close" to each other, then we need fewer draws from V^2 in order for the mean to converge to $\|\mathbf{x}\|$ than if v_i^2 are "far" apart.

BCD random projections work by *deterministically* choosing the columns of our projection matrix R, with the goal to minimize the variance due to cross terms.

Consider how we computed our variance to find "bad" $\mathbf{x}$ on Page 3.1.1. While most cross terms cancel out in expectation when using Lemma B.0.1 in computing the first moment, they do not cancel out when computing second and fourth moments, and hence they contribute to the variance.

Ideally, we would want to reduce the number of cross terms in v_i^2, which may lead to a reduction in the variance. Such a reduction of cross terms can be done by making the columns of R dependent on each other.

Furthermore, since the variance (mostly) depends on x, and not on the random variables when we choose $r_{ij} \sim SB(1)$, then re-arranging x (alternatively choosing r_{ij}) might lead to a lower variance. To put this in perspective, consider Equation 3.37 where the cross terms x_1x_3 and x_2x_4 were left. If these cross terms (in general) were large compared to other cross terms when looking at different permutations of the Hadamard matrix, then our estimates would be far away from our actual values. On the other hand, if the cross terms x_1x_3 and x_2x_4 were

small compared to the other cross terms, then we would gain a reduced variance.

We first look at the construction of R for the block and correlated part by choosing two parameters, a *block* parameter and a *correlation* parameter. We set $R = R(\beta, \rho)$ with β being the value of the block parameter, and ρ being the value of the correlation parameter. r_{ij} is drawn i.i.d. from the Sparse Bernoulli distribution with parameter $s = 1$. The intuition here is to look at groups of columns. In the ordinary case, each individual column is an estimate of $\frac{1}{k}$ of the respective squared Euclidean distances, but we now look at groups instead. Hence, we would evaluate

$$V = \sqrt{\beta} X R(\beta, \rho) \tag{3.55}$$

We give three examples to show the meaning of a *block* parameter and a *correlation* parameter. In each of these three cases, we let $p = 6$.

Choosing a block of β means we create a group of β columns, where each column has either $\lfloor \frac{p}{\beta} \rfloor$ or $\lfloor \frac{p}{\beta} \rfloor + 1$ entries, with the rest being zero.

Choosing a correlation parameter of ρ means that we create a group of ρ correlated columns, such that the majority of cross terms disappear when we calculate the respective v_{ij}^2. Suppose we denote $\tilde{r} = (r_{11} \dots r_{61})^T$. Then we would compute the Hadamard product:

$$R(1,4) = \left(\begin{array}{c|c|c|c} \tilde{r} & \tilde{r} & \tilde{r} & \tilde{r} \end{array} \right) \circ \left(\begin{array}{c|c|c|c} H_{1:6,1} & H_{1:6,2} & H_{1:6,3} & H_{1:6,4} \end{array} \right) \tag{3.56}$$

where $H_{1:6,j}$ are subsets from the Hadamard matrix. In general, we would choose ρ to be a power of 2.

Finally, for any $R(\beta, \rho)$, we look at groups of $(\beta \times \rho)$ columns for an estimate

Table 3.1: Example of BCD Random Projections

	Under Sparse Random Projections	Under BCD Random Projections
$R(4,1)$	$\begin{pmatrix} r_{11} \\ r_{21} \\ r_{31} \\ r_{41} \\ r_{51} \\ r_{61} \end{pmatrix}$	$\begin{pmatrix} r_{11} & 0 & 0 & 0 \\ 0 & r_{21} & 0 & 0 \\ 0 & 0 & r_{31} & 0 \\ 0 & 0 & r_{41} & 0 \\ 0 & 0 & 0 & r_{51} \\ 0 & 0 & 0 & r_{61} \end{pmatrix}$
$R(1,4)$	$\begin{pmatrix} r_{11} \\ r_{21} \\ r_{31} \\ r_{41} \\ r_{51} \\ r_{61} \end{pmatrix}$	$\begin{pmatrix} r_{11} & r_{11} & r_{11} & r_{11} \\ r_{21} & -r_{21} & r_{21} & -r_{21} \\ r_{31} & r_{31} & -r_{31} & -r_{31} \\ r_{41} & -r_{41} & -r_{41} & r_{41} \\ r_{51} & r_{51} & r_{51} & r_{51} \\ r_{61} & -r_{61} & r_{61} & -r_{61} \end{pmatrix}$
$R(2,2)$	$\begin{pmatrix} r_{11} \\ r_{21} \\ r_{31} \\ r_{41} \\ r_{51} \\ r_{61} \end{pmatrix}$	$\begin{pmatrix} r_{11} & 0 & -r_{11} & 0 \\ r_{21} & 0 & r_{21} & 0 \\ r_{31} & 0 & -r_{31} & 0 \\ 0 & r_{41} & 0 & -r_{41} \\ 0 & r_{51} & 0 & r_{51} \\ 0 & r_{61} & 0 & -r_{61} \end{pmatrix}$

of our squared Euclidean distances, so if $R \in \mathbb{R}^{p \times k}$, we will have $\frac{k}{\beta\rho}$ of these groups of columns. In other words, for ρ correlated columns, we extend this to $\rho\beta$ columns where each column has either $\lfloor \frac{p}{\beta} \rfloor$ or $\lfloor \frac{p}{\beta} \rfloor + 1$ entries, with the rest being zero.

$R(1, 1)$ is the (ordinary) sparse random projection.

Therefore, if $p = 6$, and we wanted to construct a matrix of the form $R(2, 2)$ with 8 columns, we would tile R with independent matrices of the form $R(2, 2)$, i.e.

$$R = \begin{pmatrix} r_{11} & 0 & -r_{11} & 0 & r_{12} & 0 & -r_{12} & 0 \\ r_{21} & 0 & r_{21} & 0 & r_{22} & 0 & r_{22} & 0 \\ r_{31} & 0 & -r_{31} & 0 & r_{32} & 0 & -r_{32} & 0 \\ 0 & r_{41} & 0 & -r_{41} & 0 & r_{42} & 0 & -r_{42} \\ 0 & r_{51} & 0 & r_{51} & 0 & r_{52} & 0 & r_{52} \\ 0 & r_{61} & 0 & -r_{61} & 0 & r_{62} & 0 & -r_{62} \end{pmatrix} \tag{3.57}$$

We can think of this as the elimination of cross terms. If we look at $R(B, 1)$, where we choose B blocks, the cross terms within each block are counted, whereas cross terms across blocks disappear. On the other hand, if we look at $R(1, C)$, the Hadamard matrix ensures that we can get rid of some cross terms. Thus, $R(B, C)$ would allow us to only have cross terms within blocks, as well as remove some cross terms in these blocks themselves.

As a comparison, if we want to estimate $\|\mathbf{x}_i\|^2$, and were given the vector $\mathbf{v}_i = (v_1, v_2, \ldots, v_k)$, then:

1. Under ordinary random projections, $v_i^2 \sim V^2_{\text{ordinary RP}}$ is an independent estimate of $\|\mathbf{x}_i\|^2$.

2. Under the SRHT, $v_i^2 \sim V_{\text{SRHT}}^2$ is an independent estimate of $\|\mathbf{x}_i\|^2$, and the values which v_{SRHT}^2 can take is strictly contained within the values which $v_{\text{ordinary RP}}^2$ can take.
3. Under the hashing trick, $v_i^2 \sim V_{\text{hash}}$ is an independent estimate of $\|\mathbf{x}_i\|^2$, but the values which v_{hash}^2 can take shares no common values with v_{SRHT}^2 or $v_{\text{ordinary RP}}^2$.
4. Under BCD random projections, each group $\sum_{j=1}^{\rho\beta} v_{l+j}^2$, where $l = 0, \rho\beta - 1, 2\rho\beta - 1, \ldots$ is an independent estimate of $\|\mathbf{x}_i\|^2$, and the values which our v_{BCD}^2 can take shares some values with $v_{\text{ordinary RP}}^2$.

3.2.1 Variance Versus Probability Bounds

We have shown in Chapter 2 that we only need probability bounds for the norm of vectors under random projections, since we can easily derive probability bounds for the estimates of inner product and Euclidean distances between vectors. However, knowledge of the variance for each group of variables $\sum_{j=1}^{\rho\beta} v_{l+j}^2,\ l = 0, \rho\beta - 1, 2\rho\beta - 1, \ldots$ (equivalently the second moments) suffices to get the probability bounds.

3.2.2 Theoretical Variances Of BCD Random Projections

We give the overall variance of our terms here. The derivation of these variances can be found in Appendix C.

Theorem 3.2.1. For $R(B,1)$ with k columns, we have that

$$\mathrm{Var}[\|\mathbf{v}\|^2] = 4B\left(\sum_{i=0}^{B-1}\left\{\sum_{a,b\geq\frac{ip}{B}+1}^{\frac{p(i+1)}{\beta}} x_{1a}^2 x_{1b}^2\right\}\right) \tag{3.58}$$

Theorem 3.2.2. For p divisible by B, the number of cross terms left in $v_{11}^2+\ldots+v_{1k}^2$ is $\frac{1}{2}(\frac{p^2}{B}-p)$ (which are more heavily weighted).

Proof. It is easy to see that we have B equal partitions of p. Since the partitions $p_1,\ldots,p_B$ have $\frac{p}{B}$ terms each, the number of cross terms removed from $p_i p_j$ must be $\frac{p^2}{B^2}$. Since there are $\binom{B}{2}$ such $p_i p_j$, then the number of cross terms which are left are

$$\binom{p}{2} - \frac{p^2}{B^2}\binom{B}{2} = \frac{1}{2}\left(p(p-1)+\frac{(B-1)p^2}{B}\right) = \frac{1}{2}\left(\frac{p^2}{B}-p\right) \tag{3.59}$$

□

Theorem 3.2.3. For $R(1,C)$ with k columns, we have that for a group of $C = 2^m$ columns, it is possible to remove at least $\frac{C-1}{C}\binom{p}{2}$ cross terms by placing the negative signs carefully. Furthermore, we have

$$\mathrm{Var}\left[\|\mathbf{v}\|^2\right] = 4C\sum_{s,t} x_{1s}^2 x_{1t}^2 - 8\sum_{i,j}\left(\sum_{s,t} x_{1s}x_{1t}H_{si}H_{sj}H_{ti}H_{tj}\right) \tag{3.60}$$

where H_{ij} is the $(i,j)^{\text{th}}$ entry of the Hadamard matrix.

Theorem 3.2.4. For $R(\beta,\rho)$ where $\beta,\rho\geq 2$, we have that

$$\mathrm{Var}\left[\|\mathbf{v}\|^2\right] = 4C\sum_{b=1}^{B}\left(\sum_{s,t\geq B(b-1)}^{b\frac{p}{B}} x_{1s}^2 x_{1t}^2\right) - 8\sum_{b=1}^{B}\left(\sum_{i,j\geq B(b-1)}^{b\frac{p}{B}}\left(\sum_{s,t\geq B(b-1)}^{b\frac{p}{B}} x_{1s}x_{1t}H_{si}H_{sj}H_{ti}H_{tj}\right)\right) \tag{3.61}$$

From these theorems, a reduction in the number of cross terms results in a greater weight on the existing terms. It seems natural then to see if we could re-order our X by columns. If the weights of x_{ij} are advantageous, we can achieve a much reduced variance.

To do so, we borrow an idea from [11], and make use of marginal information from our data. If we are interested in the Euclidean distance of each vector, we could order the columns of X by the second (or fourth) moments of $x_{\cdot i}$. Equivalently, if we are interested in the inner product or Euclidean distance between each vector, we order the columns of X by the variance (equivalently kurtosis) of each column, as we would expect $|x_{\cdot i} - x_{\cdot j}|$ to be large if the variance (kurtosis) is large.

We now explain how to sort the columns for $R(B, C)$ in order to reduce the most number of cross terms, and have the remaining cross terms be small. Without loss of generality, assume that for our p parameters, we have

$$\text{Var}[x_1] \leq \text{Var}[x_2] \leq \ldots \leq \text{Var}[x_p] \tag{3.62}$$

and that B divides p, i.e. $\frac{p}{B}$ is an integer.

Then, we first split the columns into B groups, with

$$b_1 = \{x_1, x_{B+1}, x_{2B+1}, \ldots, x_{p-B+1}\} \tag{3.63}$$

$$b_2 = \{x_2, x_{B+2}, x_{2B+2}, \ldots, x_{p-B+2}\} \tag{3.64}$$

$$\vdots \tag{3.65}$$

$$b_B = \{x_B, x_{2B}, x_{3B}, \ldots, x_p\} \tag{3.66}$$

Each group will have an equal number of elements $\frac{p}{B}$. If B does not divide p, then we split the terms as evenly as possible, so some groups will have one more element than the rest.

We can now think of the elements in our B groups as arranged in a $B \times p$

matrix, i.e.

$$\begin{pmatrix} x_1 & x_{B+1} & x_{2B+1} & \dots & x_{p-B+1} \\ x_2 & x_{B+2} & x_{2B+2} & \dots & x_{p-B+2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_B & x_{2B} & x_{3B} & \dots & x_p \end{pmatrix} \tag{3.67}$$

Our current sorting is such that we get our columns by reading the matrix vertically down from left to right.

However, our new sorting is reading the matrix horizontally from top to bottom.

In other words, if we denote our sorted x as $\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_p$, then we have

$$\tilde{x}_1 = x_1 \tag{3.68}$$

$$\tilde{x}_2 = x_{B+1} \tag{3.69}$$

$$\tilde{x}_3 = x_{2B+1} \tag{3.70}$$

$$\vdots \tag{3.71}$$

$$\tilde{x}_p = x_p \tag{3.72}$$

Then, if we wanted the Euclidean distance, the pairing of cross terms given by any $R(\beta, \rho)$ would pair $(x_i - x_i')(x_j - x_j')$ where the variance of x_is is high and x_js is low (or vice versa). Heuristically, this would avoid having weight placed on "large" cross terms, as we want to avoid cases where both x_i and x_j have large variances.

The caveat is that this may not be an optimal sorting of the columns of X, but the sorting at least ensures that the worst possible cross term is not always a product of the $\max\{\beta, \rho\}$ biggest terms. We cannot explicitly guarantee this,

since finding the column variance (and the second moment) is a heuristic, but this seems to be at most no worse off than ordinary random projections.

We note that for most real world data, all columns are usually not identically distributed with the same mean and variance, thus it is possible to exploit this re-ordering for more accurate estimations. Of course, drastic variance reduction would then be found in un-normalized data and non binary data.

3.2.3 How To View This Variance Reduction

Under ordinary random projections for matrix R, we have the variance of the estimated norm is given by:

$$4 \sum_{s,t}^{p} x_{1s}^2 x_{1t}^2 \tag{3.73}$$

and under $R(B, 1)$, we have the variance to be:

$$4B \left(\sum_{i=0}^{B-1} \left\{ \sum_{a,b \geq \frac{ip}{B}+1}^{\frac{p(i+1)}{\beta}} x_{1a}^2 x_{1b}^2 \right\} \right) \tag{3.74}$$

We consider $\mathbf{x} \in \mathbb{R}^{100}$, and have 50 entries of $\mathbf{x}$ set to 0, and the other 50 entries of $\mathbf{x}$ set to one.

If we omit the scaling factor which is constant, the weight of the cross terms of the norm using ordinary random projections would be $\binom{50}{2} = 1225$.

If $B = 2$, then under our re-arrangement of columns, we would have 2 groups, and each group would have 25 ones. Thus, our cross terms within the group would have weight $4 \times \binom{25}{2} = 1200$, which is (slightly) less than $\binom{50}{2}$.

If $B = 10$, then under our re-arrangement of columns, we would have 10 groups, and each group would have 5 ones. The cross terms within the group would then have weight $100 \times \binom{5}{2} = 1000$.

We can see that as B increases, we would continue to have a slight variance decrease.

Things are not clear-cut with real data, but we can think of the columns with lower variances equivalent to the 0s, and columns with higher variances equal to the 1s, when we use the projection of type $R(B, 1)$.

Now, consider $R(1, C)$. The variance is given to be

$$4C \sum_{s,t} x_{1s}^2 x_{1t}^2 - 8 \sum_{i,j} \left(\sum_{s,t} x_{1s} x_{1t} H_{si} H_{sj} H_{ti} H_{tj} \right) \tag{3.75}$$

With an $\mathbf{x}$, the only advantage we would get if under our re-ordering of columns, $8 \sum_{i,j} \left(\sum_{s,t} x_{1s} x_{1t} H_{si} H_{sj} H_{ti} H_{tj} \right) > 0$, and C is as low as possible.

The same also holds under $R(B, C)$, with the variance given by

$$4C \sum_{b=1}^{B} \left(\sum_{s,t \geq B(b-1)}^{b\frac{p}{B}} x_{1s}^2 x_{1t}^2 \right) - 8 \sum_{b=1}^{B} \left(\sum_{i,j \geq B(b-1)}^{b\frac{p}{B}} \left(\sum_{s,t \geq B(b-1)}^{b\frac{p}{B}} x_{1s} x_{1t} H_{si} H_{sj} H_{ti} H_{tj} \right) \right) \tag{3.76}$$

Thus, if we want random projections of the form $R(B, C)$, picking $R(B, 2)$ is the most advantageous heuristically, but this depends on the data.

3.3 Conclusion

BCD random projections are good for only certain types of data and situations. For example, if the goal was to achieve a speed-up in the computational time, or

to always get a substantial reduction in the variance regardless of the data, then BCD random projections should not be used. However, for data where the parameters have different variances or are skewed, then BCD random projections can be of great use.

The parameters β, ρ would have to be chosen by cross-validation, and computationally, the pre-processing period would take an extremely long time as we would have to compute all actual pairwise distances, and the estimates of all actual pairwise distances for each β, ρ.

Looking at the distribution of each column's variance (or kurtosis) may give indications for which parameters to pick. Furthermore, the parameter k needs to be divisible by ρ, but ρ is a power of 2.

However, future work can be done by looking at the variance (for some parameter of interest) of a concatenation of $R(\beta_i, \rho_i)$ to form the eventual matrix R.

Overall, BCD random projections can be thought of a collection of random projection matrices, using the distribution of the data to achieve a variance reduction in estimation. Based on the data X, this is equivalent in trying to optimize:

$$C = \min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \{f(R(\beta, \rho), \mathbf{x}_i, \mathbf{x}_j) - g(\mathbf{x}_i, \mathbf{x}_j)\} \tag{3.77}$$

over parameters β, ρ.

While we have presented theoretical variances for the estimation of the Euclidean distance and the inner product, we hope that our work presents a springboard for subsequent theoretical variance calculations when using other random projection algorithms to estimate other parameters of interest.

CHAPTER 4

RECONSTRUCTION OF PRINCIPAL COMPONENTS USING RANDOM PROJECTIONS

In this chapter, we will still focus on the structure of random projection matrices, but will not focus on computing estimates of Euclidean distances and inner products. Rather, we now focus on random projections to reconstruct the principal components of a matrix, and how the structure of the random matrix can play a part.

While there are works (eg, Fowler [6]) which focus on reconstructing the first principal component, we build upon and improve the works of Anaraki and Hughes [20, 22].

We briefly describe the problem, before moving on to related work and our contributions.

Suppose we have data $\mathbf{x}_i \in \mathbb{R}^p$ coming in streams, and we store a compressed version $\mathbf{v}_i \in \mathbb{R}^k$, $k \ll p$. At some point in time when we have n observations, we want to reconstruct the covariance matrix of X, and know its principal components.

One idea is to use a random projection matrix $R_{p\times k}$ to project $\mathbf{x}_i$ to $\mathbf{v}_i$. Here, we would store the random projection matrix R in addition to our $\mathbf{v}_i$s.

Then, if we think of $V_{n\times k}$ to be a matrix of our compressed $\mathbf{v}_i$s, it seems sensible that we can re-create X by getting $\tilde{X} := VR^T$, and $\tilde{X} \approx X$.

However, while distances between vectors $\mathbf{x}_i$, $\mathbf{x}_j$ are preserved, but the structure of X is not. In fact, the principal components of X are not the same as $\tilde{X}$, as

this following example shows.

Example 4.0.1. *Projecting Down From A Bivariate Normal*

We simulate $X_{3000\times 2}$, where each row $\mathbf{x}_i$ is an i.i.d. draw from $N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0.7 \\ 0.7 & 1 \end{pmatrix}\right)$. We plot our observations X in blue, and draw a black line in the direction of our first principal component.

We generate a random matrix $R_{2\times 1}$, and compute $\tilde{X}_{3000\times 2} := XRR^T$. We plot our observations $\tilde{X}$ in red for comparison.

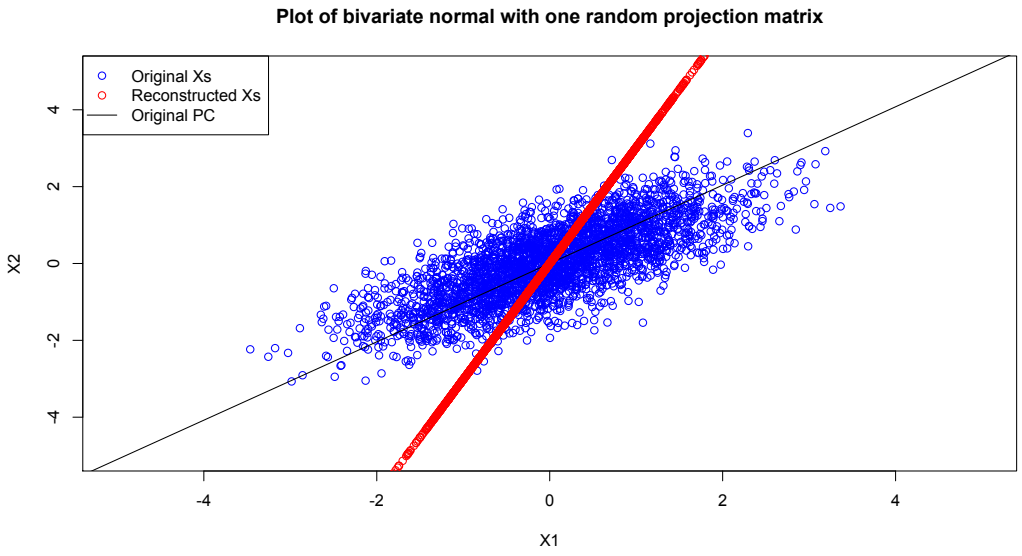

Figure 4.1: Bivariate Normal With One Projection Matrix

The result of our simulation should not be surprising, since we project our 2 dimensional points into 1 dimension (a line). Projecting our points back up into 2 dimensions would mean projecting onto a 1 dimensional subspace, which was arbitrarily chosen based on our R. Thus, we cannot just naively use one random projection matrix.

4.1 Related Work

We will use the following notation in this chapter to avoid ambiguity, as we are dealing with vector-matrix multiplication, rather than matrix-matrix multiplication. While it seems natural for $\mathbf{x}_i$ to be considered an observation (and thus a row) in a data matrix, matrix vector multiplication, eg $A\mathbf{x}$ assumes $\mathbf{x}$ to be a column vector.

Thus when dealing with vector-matrix multiplication or matrix-matrix multiplication, $\mathbf{x}_i^T$ will be considered a row vector, and $\mathbf{x}_i$ a column vector. Otherwise, we denote $\mathbf{x}_i$ to be a general observation (with no assumptions on whether it is recorded as a row or as a column).

We define $X \in \mathbb{R}^{n\times p}$ as a data matrix, with n observations $\mathbf{x}_i^T \in \mathbb{R}^{1\times p}$ having p parameters. We let $Y \in \mathbb{R}^{n\times k}$, the compressed version of our data matrix. We let C be the covariance matrix of X, given by $C = \frac{1}{n}(X-\mu)^T(X-\mu)$, with the decomposition $C = V\Sigma V^T$, and μ a $n \times p$ matrix with every row vector set to $\overline{\mathbf{x}}^T$.

Each observation $\mathbf{x}_i^T$ is assumed to be from the following probabilistic generative model:

$$\mathbf{x}_i^T = \overline{\mathbf{x}}^T + \sum_{j=1}^{d} w_{ij}\sigma_j \mathbf{v}_j^T + \mathbf{z}_i^T \tag{4.1}$$

where $\overline{\mathbf{x}}^T$ is the center of the data X, σ_j is the magnitude in each of the $\mathbf{v}_j^T$ orthonormal principal components, $\mathbf{w}_{i\cdot}^T \sim N(0, I_d)$, and $\mathbf{z}_i^T \sim N(0, \frac{\epsilon^2}{p} I_p)$.

The covariance matrix of the data X is defined to be:

$$C = \sum_{j=1}^{d} \sigma_j^2 \mathbf{v}_j \mathbf{v}_j^T \tag{4.2}$$

We don't assume $d = p$ here, since the least principal components could just be noise. The signal to noise ratio SNR is defined to be $\frac{h}{\epsilon^2}$, with $h = \sum_{j=1}^{d} \sigma_j^2$.

4.1.1 Invariance Of Principal Components Under Random Projections

A natural idea proposed by Qi [22] was to use a *different* random projection matrix for each observation $\mathbf{x}_i$.

For each observation $\mathbf{x}_i^T$, a $p \times k$ random matrix R_i is generated, with entries $r_{ij} \sim N(0, 1)$. The compressed version $\mathbf{y}_i \in \mathbb{R}^k$ is computed from $R_i^T \mathbf{x}_i$.

To get the recovered observation $\tilde{x}_i^T$, consider the following. The projection matrix P_i is given by $R_i(R_i^T R_i)^{-1} R_i^T$, and $P_i \mathbf{x}_i$ must give $\tilde{\mathbf{x}}_i$. Then it suffices to set $\tilde{\mathbf{x}}_i = R_i(R_i^T R_i)^{-1}\mathbf{y}_i$.

Under this set up, the principal components and the mean of X remain invariant under random projections, subject to a scaling quantity.

As the number of observations n goes to infinity, Qi and Hughes showed that the sample mean is estimated by:

$$\widehat{\widetilde{\mathbf{x}}} = \frac{1}{n}\frac{p}{k}\sum_{i=1}^{n} P_i \mathbf{x}_i \tag{4.3}$$

Furthermore, if $\widetilde{X}$ denoted the matrix of recovered observations $\widetilde{x}_i$, then the covariance matrix $\widetilde{C}$ of $\widetilde{X}$ gives:

$$\widetilde{C} = \widetilde{V}\widetilde{D}\widetilde{V}^T + \frac{k}{p^2}\epsilon I_p \tag{4.4}$$

where the first l columns of $\widetilde{V}$ approximates the first l eigenvectors of the actual covariance matrix C of X under some conditions.

We omit the conditions here as they can be found in the original paper, but in general the number of eigenvectors which can be reconstructed is a function of p and k.

Therefore, Equations 4.3 and 4.4 imply that we only need to store R_i and $\mathbf{y}_i$ instead of $\mathbf{x}_i$, and we can recover principal components by computing $\widehat{C}$. Furthermore, Qi also demonstrated that the accuracy in recovering principal components is better than Compressive Principal Component Analysis.

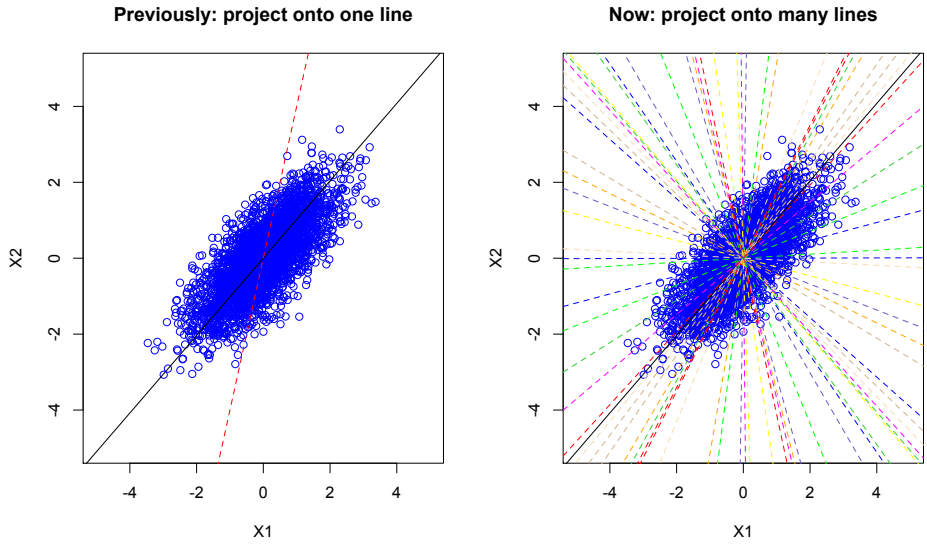

Figure 4.2: Projection Onto Multiple Lines

We give an example of how Qi's method works by extending our bivariate normal example in Example 4.0.1.

Example 4.1.1. Consider the same set up as before. Previously, using one random projection matrix R was equivalent to projecting all our observations onto

one line. Now, having one R_i for each observation $\mathbf{x}_i$ is equivalent to projecting each observation onto a different 1 dimensional subspace, as seen in Figure 4.2.

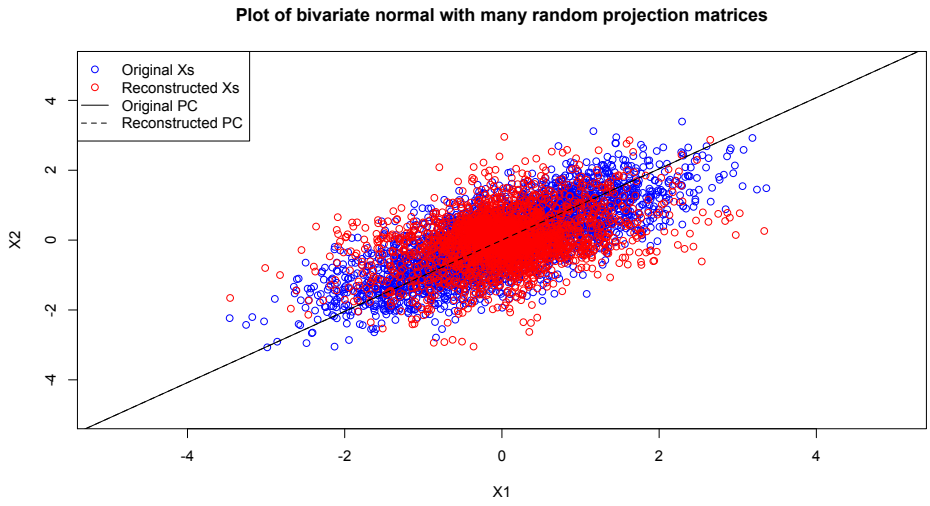

Figure 4.3: Bivariate Normal With Many Projection Matrices

When this is done, the reconstructed points have the same mass near the center, and also lie in the same direction as the original principal components.

Qi's method does not come with some drawbacks however. Although the method allows recovery of more than one principal component, there is a storage issue. Storing the compressed observations $\mathbf{y}_i$ is not a problem, but storing each R_i is costly. Furthermore, computing the projection P_i is of order $O(p^3)$ (even with LU decomposition instead of computing the inverse directly), so overall reconstruction takes $O(np^3)$.

Despite these drawbacks, Qi's method is promising. Qi approaches the problem of reconstruction of the principal components differently, and thus there is potential for this method to be improved. Similar to how the structure of random projection matrices has shifted from $r_{ij} \sim N(0,1)$ to r_{ij} drawn from Sparse

Bernoulli distributions, we can imagine doing the same here as well.

4.1.2 Principal Component Analysis Via Very Sparse Random Projections

The work of Anaraki and Hughes [20] extends the work of Qi [22], which combines compressed sensing together with random projections. Their work is also different from common random projection algorithms as they create an individual random projection matrix for each observation. We review their work briefly.

Recall that the Sparse Bernoulli distribution with parameter s is given by:

$$r = \begin{cases} \sqrt{s} & \text{with probability } \frac{1}{2s} \\ -\sqrt{s} & \text{with probability } \frac{1}{2s} \\ 0 & \text{with probability } 1 - \frac{1}{s} \end{cases} \tag{4.5}$$

We denote μ_k to be the k^{th} moment of r. We denote $h = \sum_{i=1}^{d} \sigma_j^2$, and the signal to noise ratio $\frac{1}{\text{SNR}}$ to be $\frac{\epsilon^2}{h}$. We denote $\kappa = \frac{\mu_4}{\mu_2^2} - 3$.

For each observation $\mathbf{x}_i^T$, a random matrix $R_i \in \mathbb{R}^{p \times k}$ is created, where $r_{ij} \sim \text{SB}(s)$. We denote $\mathbf{y}_i = R_i^T \mathbf{x}_i$, and $\widetilde{\mathbf{x}}_i = R_i R_i^T \mathbf{x}_i$. As the number of observations n goes to infinity, the sample mean of all observations is estimated by

$$\widehat{\overline{\mathbf{x}}} = \frac{1}{nk\mu_2} \sum_{i=1}^{n} R_i R_i^T \mathbf{x}_i \tag{4.6}$$

with variance

$$\text{Var}[\widehat{\overline{\mathbf{x}}}] = \frac{p}{kn} \left(h \left(1 + \frac{1}{\text{SNR}} \right) \left(\frac{k + \kappa + 1 + p}{p} \right) + \left(\frac{p + \kappa + 1}{p} \right) \|\overline{\mathbf{x}}\|_2^2 \right) \tag{4.7}$$

Furthermore, if $\widetilde{X}$ denoted the matrix of recovered observations $\widetilde{x_i}$, then the covariance matrix $\widetilde{C}$ of $\widetilde{X}$ gives:

$$\widehat{C} = \frac{1}{(k^2+k)\mu_2^2}\frac{1}{n}\sum_{i=1}^{n} R_i R_i^T(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T R_i R_i^T \tag{4.8}$$

$$= C_{\text{true}} + \alpha I_n + E \tag{4.9}$$

with $\alpha := \frac{h}{k+1} + \left(\frac{\kappa}{p(k+1)} + \frac{k+p+1}{p(k+1)}\right)\epsilon^2$ and $E = \frac{\kappa}{k+1}\sum_{j=1}^{d} \text{diag}(\mathbf{v}_j^T \mathbf{v}_j)$.

E here can be seen as some perturbation term comprised of the first few eigenvectors. As the number of observation increases, then $\alpha \to 0$. Thus $\widehat{C}$ asymptotically has the same eigenvectors as C_{true}, but perturbed eigenvalues.

The results of Anaraki and Hughes [20] give rise to the following observations. For R_i with $r_{ij} \sim \text{SB}(s)$, on average $\frac{s-1}{s}$ of its entries will be zeros. Therefore, if n is large and $\mathbf{x}$ is centered, by aggressively increasing s, we only need to store each $\widetilde{\mathbf{x}}_i = R_i R_i^T \mathbf{x}_i$ and $\mathbf{y}_i = R_i^T \mathbf{x}_i$, instead of storing R_i and $\mathbf{y}_i$.

Secondly, since $\widetilde{\mathbf{x}}_i$ is sparse compared to $\mathbf{x}_i$, then performing SVD on $\widetilde{X}$ to recover principal components is much faster compared to performing SVD on the original X.

While Anaraki and Hughes mitigated some of the drawbacks in the work by Qi and Hughes, their work can be further improved and extended. While increasing the parameter s does lead to reduced storage cost, the variance in estimating the principal components is necessarily higher. Furthermore, if s is too high, then the reconstruction is equivalent to selecting k' parameters at random to be non zero for each entry $\mathbf{x}_i^T$, and zeroing out the other $p - k'$ entries, which can be done without random matrices.

For example, a sparsity parameter of $s = \frac{1}{p}$ using k columns is equivalent (in

expectation) to zeroing out $p - k$ entries for each $\mathbf{x}_i^T$.

On the other hand, if s was lower, then we would have greater accuracy, but the storage costs of R_i and $\mathbf{y}_i$ would exceed the original storage cost of $\mathbf{x}_i$.

We thus propose a new method involving Hadamard type random matrices with correlated r_{ij} to reconstruct the principal components.

4.2 Our Contributions: Semi-Deterministic Random Projections

Consider an example of the set up behind random projection algorithms involving a symmetric matrix $X_{p\times p}$ as a whole. Given the eigendecomposition of X as PDP^T, an orthogonal random projection matrix $R_{p\times k}$ is formed, with $RR^T \approx I_p$ and the following matrix product

$$\Sigma := R^T X R \tag{4.10}$$

is computed. If we wanted an estimate of the top k eigenvectors of X, then it suffices to compute the eigendecomposition of $\Sigma := QD_2Q^{-1}$, and then compute RQ to get an estimate of the top k eigenvectors of X.

4.2.1 Random Projection Structure

With this as a basis, we propose using a strategy similar to the Subsampled Randomized Hadamard Transform [4] to construct R. We will call our construction Semi-Deterministic Random Projections (SDP for short). Here, we will assume

p is a power of 2. We will show the case where p is not a power of 2 in Chapter 4.2.3.

We recall that the Hadamard matrix can be defined recursively as

$$H_0 = [1] \tag{4.11}$$

$$H_k = \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{bmatrix} \tag{4.12}$$

Our construction is as follows:

for *each incoming* $\mathbf{x}_i \in \mathbb{R}^p$ **do**
  Construct the Hadamard matrix $\tilde{H} := H_p$, and take the subset $H := \tilde{H}_{1:p,1:k}$.
  Let $D_{p\times p}$ be a diagonal matrix, with $d_{ii} \sim \{\pm 1\}$ with equal probability.
  Set $R_i := DH$.
  Compute $\mathbf{y}_i = R^T \mathbf{x}_i$
  Store $\mathbf{y}_i$, diag(R_i)
end

Figure 4.4: Algorithm For Semi-Deterministic Projections

The key idea here is that instead of storing a matrix R_i (with either dense entries as in Qi, or {0,1} entries as in Anaraki), we are only storing a vector $\mathbf{r}_i$ with ±1 entries. This reduces the storage space substantially. Furthermore, the matrix R_i can be constructed recursively without the full construction of the Hadamard matrix.

SDP differs from the Subsampled Randomized Hadamard Transform in two ways. Firstly, we do not let H be a $p \times p$ matrix, but instead be a $p \times k$ matrix, and secondly, we do not pick columns without replacement from H.

We do so because in this case, an individual observation would be a product of matrix multiplications as in Equation 4.10. Conversely, if we were looking

at the original random projection case with $V = XR$, an individual observation would be a column, and thus we would need to pick columns with replacement.

Lemma 4.2.1. Such an $R_{p\times k}$ from our construction has these properties

1. $\mathbb{E}[RR^T] = kI_p$
2. $\mathbb{E}[RR^TRR^T] = kpI_p$
3. The diagonal entries of RR^T and RR^TRR^T are always exact, regardless of the entries in R.
4. Suppose we let $P = RR^T$, and $Q = HH^T$, where $H := \tilde{H}_{1:p,1:k}$.
   (a) If $Q_{ij} = 0$, then $P_{ij} = 0$
   (b) If $Q_{ij} = q \neq 0$, then $P_{ij} \sim \{\pm q\}$ with equal probability

Proof. We denote the first column of R to be

$$R_{\cdot i} = (r_1, r_2, \ldots, r_p)^T \tag{4.13}$$

For $r \sim \{\pm 1\}$ with equal probabilities

$$\mathbb{E}[r^{2k}] = 1 \quad \mathbb{E}[r^{2k+1}] = 0 \tag{4.14}$$

This follows from the fact that r is symmetric about zero, and thus odd moments must have expectation 0. Furthermore, since r only takes values +1 and −1 with equal probability, then $\mathbb{E}[r^{2k}] = \frac{1}{2}\left(1^{2k} + (-1)^{2k}\right) = 1$.

Let $H := \tilde{H}_{1:p,1:p}$ be the subset of our Hadamard matrix.

Denoting $A = RR^T$, we can write the $(i, j)^{\text{th}}$ term of A as

$$A_{ij} = \sum_{m=1}^{k} r_i r_j H_{mi} H_{mj} \tag{4.15}$$

We now look at the expectation of diagonal terms, and off-diagonal terms. For $i = j$, we have

$$\mathbb{E}[A_{ii}] = \mathbb{E}\left[\sum_{m=1}^{k} r_i^2 H_{1i}^2\right] \tag{4.16}$$

$$= \mathbb{E}\left[\sum_{m=1}^{k} r_i^2\right] \qquad \text{since } H_{i1}^2 = 1 \tag{4.17}$$

$$= \sum_{m=1}^{k} \mathbb{E}[r_i^2] \tag{4.18}$$

$$= k \tag{4.19}$$

For $i \neq j$, we have

$$\mathbb{E}[A_{ij}] = \mathbb{E}\left[\sum_{m=1}^{k} r_i r_j H_{mi} H_{mj}\right] \tag{4.20}$$

$$= \sum_{m=1}^{k} \mathbb{E}[r_i]\mathbb{E}[r_j] H_{mi} H_{mj} \tag{4.21}$$

$$= \mathbb{E}[r_i]\mathbb{E}[r_j] \sum_{m=1}^{k} H_{mi} H_{mj} \tag{4.22}$$

$$= 0 \tag{4.23}$$

Thus, we have must have $\mathbb{E}[RR^T] = \mathbb{E}[A] = kI_p$.

Furthermore, if we do not take expectations of A_{ij}, then

$$A_{ij} = r_i r_j \sum_{m=1}^{k} \tilde{H}_{mi} \tilde{H}_{mj} \tag{4.24}$$

Since r_i, r_j are independent, then their product must either be 1 or -1 with equal probability, and thus statement 4) holds.

We now denote $B = RR^T RR^T = A^2$, and note that since $(RR^T)^T = RR^T$, A must be symmetric.

Let $\mathbf{a}_i^T$ denote the i^{th} row of A. By symmetry, we must have each $(i, j)^{\text{th}}$ entry of B to be $B_{ij} = \mathbf{a}_i^T \mathbf{a}_i$. Similar to the above, we compute the expectation of the diagonal terms and off diagonal terms of B.

Consider each j^{th} entry in the i^{th} row $\mathbf{a}_i$. When $i = j$, then

$$a_{ii} = \sum_{m=1}^{k} r_i^2 H_{mi}^2 \tag{4.25}$$

$$= r_i^2 \sum_{m=1}^{k} H_{mi}^2 \tag{4.26}$$

$$= \sum_{m=1}^{k} H_{mi}^2 \tag{4.27}$$

$$= k \tag{4.28}$$

When $i \neq j$, then

$$a_{ij} = \sum_{m=1}^{k} r_i r_j H_{mi} H_{mj} \tag{4.29}$$

$$= r_i r_j \sum_{m=1}^{k} H_{mi} H_{mj} \tag{4.30}$$

We now consider the off diagonal terms of B, which are given by $\mathbf{a}_i^T \mathbf{a}_j,\ i \neq j$.

$$\mathbb{E}[B_{ij}] = \mathbb{E}\left[\mathbf{a}_i \mathbf{a}_j^T\right] \tag{4.31}$$

$$= \mathbb{E}\left[\sum_{l=1}^{p} a_{il} a_{jl}\right] \tag{4.32}$$

$$= \mathbb{E}\left[\sum_{l=1}^{p} \left(r_i r_j r_l^2 \left\{ \sum_{m=1}^{k} H_{mi} H_{ml} \right\} \left\{ \sum_{m=1}^{k} H_{mj} H_{ml} \right\} \right)\right] \tag{4.33}$$

$$= 0 \tag{4.34}$$

since $i, j \neq l$ at each index, thus $\mathbb{E}[r_i r_j r_l^2] = 0$.

We now consider the diagonal terms of B, and drop the expectation sign.

The reason why we can do this is due to the fact that

$$B_{ii} = \mathbf{a}_i^T \mathbf{a_i} \tag{4.35}$$

$$= \sum_{l=1}^{p} a_{il}^2 \tag{4.36}$$

$$= \sum_{l=1}^{p} r_l^2 r_i^2 \left(\sum_{m=1}^{k} H_{mi} H_{ml} \right)^2 \tag{4.37}$$

$$= \sum_{l=1}^{p} \left(\sum_{m=1}^{k} H_{mi} H_{ml} \right)^2 \tag{4.38}$$

which isn't random at all. Now, let $\widetilde{H}$ be a subset $\widetilde{H} = H_{1:p,1:k}$ of our Hadamard matrix H, then the *diagonal* elements of $\widetilde{H}\widetilde{H}^T\widetilde{H}\widetilde{H}^T$ are identical to the diagonal elements of $B := RR^TRR^T$. It suffices to find the diagonal elements of $\widetilde{H}\widetilde{H}^T\widetilde{H}\widetilde{H}^T$.

We first note that the diagonal elements of $\widetilde{H}\widetilde{H}^T\widetilde{H}\widetilde{H}^T$ are equal to the diagonal elements of $\sum_{j=1}^{k}(H_{.j}H_{.j}^T)^2$.

Then, since $H_{.i}$ is a vector of ±1s, if we let $C = (H_{.j}H_{.j}^T)$ for any j, then C must be a matrix consisting of ±1s. Furthermore, the diagonal elements $(C^2)_{ii}$ must be equal to $\sum_{j=1}^{p} C_{ij}^2$, which is a sum of (positive) ones, and therefore is equal to p.

This implies that each diagonal element of $\sum_{j=1}^{k}(H_{.j}H_{.j})^2$ is equal to kp, which also must be equal to the corresponding diagonal element of $\widetilde{H}\widetilde{H}^T\widetilde{H}\widetilde{H}^T$.

Thus, we have that $\mathbb{E}[RR^TRR^T] = kpI_p$. □

Having exact diagonal entries for RR^T, $(RR^T)(RR^T)$ would mean greater accuracy in matrix reconstruction. Furthermore, when constructing R with our method, we would want to choose k which factorizes into more powers of 2, in order to control the number of zeros in the off-diagonal entries of RR^T.

4.2.2 Theoretical Bounds And Guarantees

Theorem 4.2.1. Using SDP, the estimate of the sample mean when the number of observations n tends to infinity is given by:

$$\widehat{\overline{\mathbf{x}}} = \frac{1}{nk} \sum_{i=1}^{n} R_i R_i^T \mathbf{x}_i \tag{4.39}$$

with an upper bound on the variance:

$$\mathrm{Var}[\widehat{\overline{\mathbf{x}}}] \leq \frac{1}{n} \left(h \left(1 + \frac{p}{k} \frac{1}{\mathrm{SNR}} + k^2 \right) + \frac{p-k}{k} ||\overline{\mathbf{x}}||_2^2 \right) \tag{4.40}$$

where $h = \sum_{j=1}^{d} \sigma_j^2$, with σ_j^2 being the magnitude of the j^{th} principal component.

Proof. Equation 4.39 follows from Lemma 4.2.1.

The variance of the estimator $\widehat{\overline{\mathbf{x}}}$ in Equation 4.40 follows mostly the same steps as Anaraki and Hughes, and only differs when substituting values of (different) expected values from Lemma 4.2.1. Thus, we only show this proof for completion. We write:

$$\begin{aligned}
\mathrm{Var}[\widehat{\overline{\mathbf{x}}}] &= \mathbb{E}\left[||\widehat{\overline{\mathbf{x}}} - \overline{\mathbf{x}}||_2^2 \right] && (4.41) \\
&= \frac{1}{n^2} \mathrm{tr} \left(\mathbb{E} \left[\sum_{i=1}^{n} \left(\frac{1}{k} R_i R_i^T \mathbf{x}_i - \overline{\mathbf{x}} \right) \left(\frac{1}{k} R_i R_i^T \mathbf{x}_i - \overline{\mathbf{x}} \right)^T \right] \right) && (4.42) \\
&= \frac{1}{n} \mathrm{tr} \left(\mathbb{E} \left[\left(\frac{1}{k} R_i R_i^T \mathbf{x}_i - \overline{\mathbf{x}} \right) \left(\frac{1}{k} R_i R_i^T \mathbf{x}_i - \overline{\mathbf{x}} \right)^T \right] \right) && (4.43) \\
&= \frac{1}{n} \frac{1}{k^2} \mathrm{tr} \left(\mathbb{E} \left[RR^T (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^T RR^T \right] \right) && (4.44) \\
&\quad + \frac{1}{n} \mathrm{tr} \left(\mathbb{E} \left[\left(\frac{1}{k} RR^T \overline{\mathbf{x}} - \overline{\mathbf{x}} \right) \left(\frac{1}{k} RR^T \overline{\mathbf{x}} - \overline{\mathbf{x}} \right)^T \right] \right)
\end{aligned}$$

We look at the first term in the above equation. WLOG, suppose $\mathbf{x}$ is centered. We write $h = \sum_{j=1}^{d} \sigma_j^2$. Since the trace only deals with the diagonal elements of a

matrix, we have:

$$\frac{1}{n}\frac{1}{k^2}\text{tr}\left(\mathbb{E}\left[RR^T(\mathbf{x}-\overline{\mathbf{x}})(\mathbf{x}-\overline{\mathbf{x}})^T RR^T\right]\right) \tag{4.45}$$

$$\leq \frac{1}{nk^2}\left(k^2\left(\text{tr}\left(C_{\text{true}}+\frac{\epsilon^2}{k}I_p\right)+k^2\sum_{i=1}^{p}\sum_{j=1}^{d}\sigma_j^2 v_{j,i}^2\right)\right) \tag{4.46}$$

$$= \frac{1}{n}\left(h+\frac{p\epsilon^2}{k}+k^2 h\right) \tag{4.47}$$

$$= \frac{h}{n}\left(1+\frac{p}{k}\frac{1}{\text{SNR}}+k^2\right) \tag{4.48}$$

Using Lemma 4.2.1, we can write the second term in the above equation as

$$\frac{1}{n}\text{tr}\left(\mathbb{E}\left[\left(\frac{1}{k}RR^T\overline{\mathbf{x}}-\overline{\mathbf{x}}\right)\left(\frac{1}{k}RR^T\overline{\mathbf{x}}-\overline{\mathbf{x}}\right)^T\right]\right) = \frac{1}{n}\mathbb{E}\left[\left\|\frac{1}{k}RR^T\overline{\mathbf{x}}-\overline{\mathbf{x}}\right\|_2^2\right] \tag{4.49}$$

$$= \frac{1}{nk^2}\overline{\mathbf{x}}\,\mathbb{E}[RR^T RR^T]\,\overline{\mathbf{x}}^T - \|\overline{\mathbf{x}}\|_2^2 \tag{4.50}$$

$$= \frac{p-k}{nk}\|\overline{\mathbf{x}}\|_2^2 \tag{4.51}$$

Therefore, we have

$$\text{Var}[\widehat{\overline{\mathbf{x}}}] \leq \frac{h}{n}\left(1+\frac{p}{k}\frac{1}{\text{SNR}}+k^2\right)+\frac{p-k}{nk}\|\overline{\mathbf{x}}\|_2^2 \tag{4.52}$$

□

Suppose we let $H := \tilde{H}_{1:p,1:k}$, which is a subset of the first k columns of the Hadamard matrix of size p as stated before. Let $K_{p\times p}$ be a matrix where $K_{ii} = 0$ and $K_{ij} = k^2,\ i \neq j$.

Theorem 4.2.2. For our semi-deterministic matrices, the estimate of the sample covariance matrix is given by

$$\widehat{C} = \frac{1}{K+(HH^T \circ HH^T)}\sum_{i=1}^{n} R_i R_i^T \mathbf{x}_i \mathbf{x}_i^T R_i R_i^T \tag{4.53}$$

where we do elementwise division of matrices (with some abuse of notation). This can be decomposed to

$$\widehat{C} = \widehat{C}_{\text{true}} + E \tag{4.54}$$

where $\widehat{C}_{\text{true}} = C_{\text{true}} + \frac{\epsilon^2}{k} I_p$, and E a diagonal matrix where $E_{ii} = \frac{\sum_{j\neq i}^{p} \sigma_j^2 v_j^2 \left(\sum_{m=1}^{k} H_{mj} H_{mi}\right)^2}{k^2}$, with H_{ij} denoting the $(i, j)^{\text{th}}$ term of the Hadamard matrix H.

Proof. We write:

$$\mathbb{E}[RR^T \mathbf{x}\mathbf{x}^T RR^T] = \sum_{i=1}^{d} \sigma_i^2 \mathbb{E}[RR^T \mathbf{v}_i \mathbf{v}_i^T RR^T] + \mathbb{E}[RR^T \mathbf{z}\mathbf{z}^T RR^T] \tag{4.55}$$

Consider an arbitrary principal component $\mathbf{v}_j$. Denoting A to be $\mathbb{E}[RR^T \mathbf{v}_i \mathbf{v}_i^T RR^T]$, and now letting v_j be the j^{th} component of $\mathbf{v}_i$, we have that

$$A_{ii} = \sum_{j=1}^{p} v_j^2 \left(\sum_{m=1}^{k} H_{mi} H_{mj}\right)^2 \tag{4.56}$$

$$= k^2 v_i^2 + \sum_{j\neq i}^{p} v_j \left(\sum_{m=1}^{k} H_{mi} H_{mj}\right)^2 \tag{4.57}$$

and

$$A_{ij} = v_i v_j \left(\left\{ \sum_{m=1}^{k} H_{mi} H_{mi} \right\} \left\{ \sum_{m=1}^{k} H_{mj} H_{mj} \right\} + \left\{ \sum_{m=1}^{k} H_{mi} H_{mj} \right\}^2 \right) \tag{4.58}$$

$$= v_i v_j \left(k^2 + \left\{ \sum_{m=1}^{k} H_{mi} H_{mj} \right\}^2 \right) \tag{4.59}$$

Then, letting K be the $p \times p$ matrix with off diagonal entries k^2, and diagonal entries of 0, and writing $H = \tilde{H}_{1:p,1:k}$, we have:

$$A = \mathbf{v}_i^T \mathbf{v}_i \circ (K + (HH^T \circ \widetilde{H}\widetilde{H}^T)) + E \tag{4.60}$$

where $E_{ii} = \sum_{j\neq i}^{p} v_j^2 \left(\sum_{m=1}^{k} H_{mj} H_{mi}\right)^2$ and $E_{ij} = 0$ for $i \neq j$.

With Lemma 4.2.1, we can express:

$$\mathbb{E}[RR^T\mathbf{z}\mathbf{z}^TRR^T] = \frac{\epsilon^2}{p}\mathbb{E}[RR^TRR^T] \tag{4.61}$$

$$= k\epsilon^2 I_p \tag{4.62}$$

Thus:

$$\frac{\mathbb{E}[RR^T\mathbf{x}\mathbf{x}^TRR^T]}{K + (\widetilde{H}\widetilde{H}^T \circ \widetilde{H}\widetilde{H}^T)} = \sum_{i=1}^{d} \sigma_i^2 \mathbf{v}_i^T\mathbf{v}_i + \frac{\epsilon^2}{k}I_p + E \tag{4.63}$$

(again with some slight abuse of notation for elementwise division) where $E_{ii} = \frac{\sum_{j\neq i}^{p}\sigma_j^2 v_j^2\left(\sum_{m=1}^{k} H_{mj}H_{mi}\right)^2}{k^2}$, and $E_{ij} = 0$ for $i \neq j$. □

Lemma 4.2.2. With our SDP construction of R, we have:

$$\mathbb{E}\left[\|RR^T\mathbf{x}\|_2^4\right] \leq (kp)^2\left(\sum_{i=1}^{p} x_i^4 + 4\sum_{i=1}^{p-1}\sum_{j>i}^{p} x_i^2x_j^2\right) \tag{4.64}$$

Proof. Suppose we denote $\mathbf{x}_i$ to be the i^{th} entry of $\mathbf{x}$, and we let $\mathbf{a}_i$ denote the i^{th} row of RR^T. We use $\mathbf{a}_i$ interchangeably to denote the i^{th} column since RR^T is symmetric.

In the proof of Lemma 4.2.1, we have shown that for $i \neq j$, $\mathbb{E}\left[\mathbf{a}_i^T\mathbf{a}_j\right] = 0$, which implies $\mathbb{E}\left[\mathbf{a}_i^T\mathbf{a}_j\mathbf{a}_i^T\mathbf{a}_l\right] = 0$ for $j \neq l$. We have also shown that $\mathbf{a}_i^T\mathbf{a}_i = kp$, which is exact.

From [20], the expansion of $\|RR^T\mathbf{x}\|_2^4$ is given by:

$$\|RR^T\mathbf{x}^T\|_2^4 = \sum_{i=1}^{p}\sum_{j=1}^{p}\sum_{m=1}^{p}\sum_{n=1}^{p} x_i x_j x_m x_n \mathbf{a}_i^T\mathbf{a}_j\mathbf{a}_m^T\mathbf{a}_n \tag{4.65}$$

$$\begin{aligned}
&= \sum_{i=1}^{p} x_i^4(\mathbf{a}_i^T\mathbf{a}_i)^2 + 4\sum_{i=1}^{p}\sum_{j\neq i} x_i^3 x_j \mathbf{a}_i^T\mathbf{a}_i\mathbf{a}_i^T\mathbf{a}_j \\
&\quad + \sum_{i=1}^{p}\sum_{j\neq i} x_i^2 x_j^2 \mathbf{a}_i^T\mathbf{a}_i\mathbf{a}_j^T\mathbf{a}_j + 2\sum_{i=1}^{p}\sum_{j\neq i} x_i^2 x_j^2(\mathbf{a}_i^T\mathbf{a}_j)^2 \\
&\quad + 2\sum_{i=1}^{p}\sum_{j\neq i}\sum_{m\neq i,j} x_i^2 x_j x_m \mathbf{a}_i^T\mathbf{a}_i\mathbf{a}_j^T\mathbf{a}_m \\
&\quad + 4\sum_{i=1}^{p}\sum_{j\neq i}\sum_{m\neq i,j} x_i^2 x_j x_m \mathbf{a}_i^T\mathbf{a}_j\mathbf{a}_i^T\mathbf{a}_m \\
&\quad + \sum_{i=1}^{p}\sum_{j\neq i}\sum_{m\neq i,j}\sum_{n\neq i,j,m} x_i x_j x_m x_n \mathbf{a}_i^T\mathbf{a}_j\mathbf{a}_m^T\mathbf{a}_n
\end{aligned} \tag{4.66}$$

Taking expectations, it follows that:

$$\mathbb{E}\left[\|RR^T\mathbf{x}\|_2^4\right] = \sum_{i=1}^{p} x_i^4(\mathbf{a}_i^T\mathbf{a}_i)^2 + \sum_{i=1}^{p}\sum_{j\neq i} x_i^2 x_j^2 \mathbf{a}_i^T\mathbf{a}_i\mathbf{a}_j^T\mathbf{a}_j \tag{4.67}$$

$$+ 2\sum_{i=1}^{p}\sum_{j\neq i} x_i^2 x_j^2 \mathbb{E}[(\mathbf{a}_i^T\mathbf{a}_j)^2] \tag{4.68}$$

$$= (kp)^2\left(\sum_{i=1}^{p} x_i^4 + 2\sum_{i=1}^{p-1}\sum_{j>i}^{p} x_i^2 x_j^2\right) \tag{4.69}$$

$$+ 2\sum_{i=1}^{p-1}\sum_{j>i}^{p} x_i^2 x_j^2 \mathbb{E}\left[(\mathbf{a}_i^T\mathbf{a}_j)^2\right] \tag{4.70}$$

Here, we reach a snag, since the expected value of $\mathbb{E}\left[(\mathbf{a}_i^T\mathbf{a}_j)^2\right]$ depends on $H_{i.}, H_{j.}$, and hence are not all equal. However, this value is still upper bounded by $(kp)^2$, and we can write:

$$\mathbb{E}\left[\|RR^T\mathbf{x}\|_2^4\right] \leq (kp)^2\left(\sum_{i=1}^{p} x_i^4 + 4\sum_{i=1}^{p-1}\sum_{j>i}^{p} x_i^2 x_j^2\right) \tag{4.71}$$

□

We have the upper bound of our expected value is lower than the expected

value of $\|RR^T\mathbf{x}\|_2^4$ when compared to ordinary random projections with $r_{ij} \sim$ SB(1).

With Lemma 4.2.1, Lemma 4.2.2, and the E term given in Theorem 4.2.2, (upper) bounds on the error of the estimated covariance matrix from the sample covariance matrix, as well as the error on the estimated eigenvectors can immediately be found by substituting in the values of these bounds, by following the structure in [20].

4.2.3 When p Is Not A Power Of 2

We now revisit the case where p is not a power of 2. Let $\tilde{p} = 2^s$, where $\tilde{p}$ is the nearest power of 2 greater than p. A naive method would be to set $\tilde{\mathbf{x}}_i^T \in \mathbb{R}^{\tilde{p}}$ to be $[\mathbf{x}_i^T \mid \mathbf{0}_{(\tilde{p}-p)\times 1}]$, where we tile $\mathbf{x}_i^T$ with zeroes. We would then compute the respective $\tilde{\hat{\tilde{\mathbf{x}}}}$, and set $\hat{\tilde{\mathbf{x}}} = \tilde{\hat{\tilde{\mathbf{x}}}}_{1:p}$.

However (dropping the subscripts), consider how we found $\hat{\tilde{\mathbf{x}}} = RR^T\mathbf{x}$. In our tiling case, we would write:

$$\begin{pmatrix} \hat{\tilde{\mathbf{x}}}_{1:p} \\ \hat{\tilde{\mathbf{x}}}_{p+1:\tilde{p}} \end{pmatrix} = \begin{pmatrix} (RR^T)_{1:p,1:p} & (RR^T)_{1:p,(p+1):\tilde{p}} \\ (RR^T)_{(p+1):\tilde{p},1:p} & (RR^T)_{(p+1):\tilde{p},(p+1):\tilde{p}} \end{pmatrix} \begin{pmatrix} \mathbf{x}_{1:p} \\ \mathbf{0} \end{pmatrix} \tag{4.72}$$

But if we are going to set $\hat{\tilde{\mathbf{x}}}_{p+1:\tilde{p}}$ to zero after computing it, then we can write:

$$\begin{pmatrix} \hat{\tilde{\mathbf{x}}}_{1:p} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} (RR^T)_{1:p,1:p} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x}_{1:p} \\ \mathbf{0} \end{pmatrix} \tag{4.73}$$

and we can just use the subset $R_{1:p,1:p}$ from $R_{1:\tilde{p},1:\tilde{p}}$ for our matrix multiplication.

4.3 Our Experiments With Synthetic Data

We first generate synthetic data from the following model.

$$\mathbf{x}_i^T = \overline{\mathbf{x}}^T + \sum_{j=1}^{d} w_{ij}\sigma_j \mathbf{v}_j^T + \mathbf{z}_i^T \tag{4.74}$$

We randomly draw each entry of $\overline{\mathbf{x}}$ from Uniform$(0, 20)$, and generate 3 principal components. Each principal component $\mathbf{v}_j$ has every entry randomly drawn from Uniform$(0, 1)$, before being normalized to have length 1. We let $\sigma = (\sigma_1, \sigma_2, \sigma_3) = (20, 16, 13)$. We fix the number of variables $p = 500$ for each $\mathbf{x}_i$. We have $\mathbf{w}_{i\cdot}^T \sim N(0, I_d)$, and $\mathbf{z}_i^T \sim N(0, \frac{\epsilon^2}{p} I_p)$ with $\epsilon = \sqrt{p}$.

We let X denote our original data, $\tilde{X}$ denote the centered data, and Y denote our compressed data, and give a comparison of the memory needed for storage, with $n = 5000$, $p = 500$, and $k = 150$:

	X	All R_is	$\widetilde{X}$	Y
RP	19.1MB	79MB	19.1MB	5.9MB
SDP	19.1MB	540KB	19.1MB	5.9MB

Table 4.1: Comparison Of Storage Between SDP And RP For Synthetic Data

We store the data as ordinary CSV files, as we want to show that even without any data structures, the storage space by simply storing a single vector $\mathbf{r}_i$ rather than storing a matrix R_i is reduced substantially.

Under sparse random projections (RP) via Anaraki and Hughes, constructing each random projection where $r_{ij} \sim$ SB(1) would not be advantageous, regardless of whether the data is centered or not. Centered data would require storage of (19.1 MB + 5.9MB) in order to reconstruct the principal components,

which is certainly greater than 19.1MB. Uncentered data would require much more memory, with all R_is already taking up 79MB.

However, using our construction with SDP, we would store 540KB in order to reconstruct every R_i (since we only need to store the signs d_{ii}) and 5.9MB for compressed Y, with total storage space less than a third of the space needed to store X, which is about the ratio $\frac{k}{p}$. This allows us to reconstruct the principal components for both centered and uncentered data. In general, we would expect to store about $\frac{k}{p}$ of the original size of the data, since the cost of storing each R_i is negligible in comparison, thus if n is large, we could potentially allow k to be smaller.

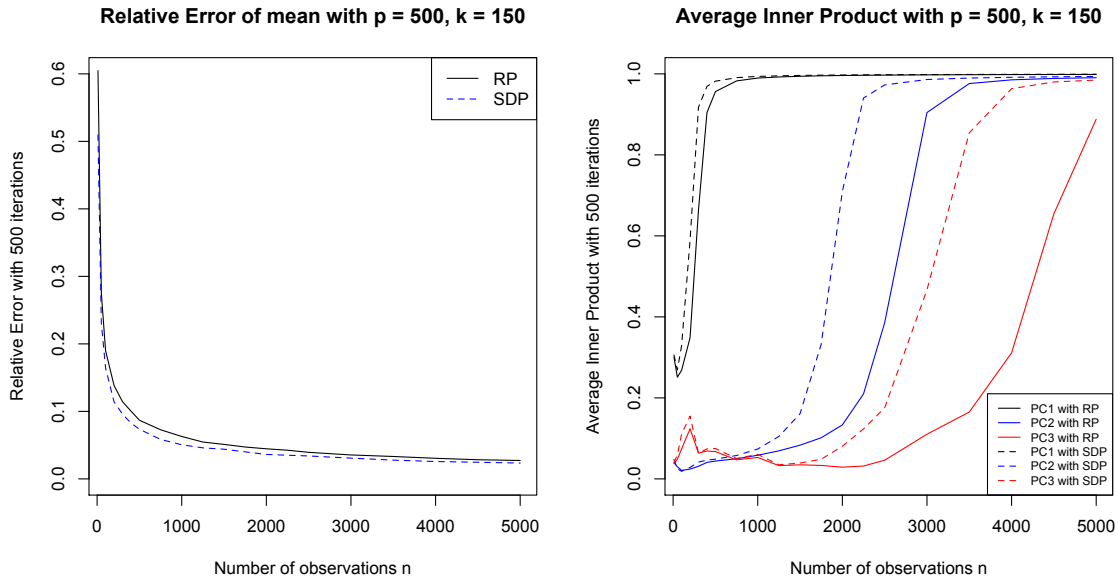

Figure 4.5: Relative Error Of Mean And Comparison Of Inner Product Using SDP And RP For Synthetic Data

We next show the comparison of the relative error of the recovered mean with simulated data, given by by $\frac{\|\bar{x}_{\text{true}} - \bar{x}_{\text{recovered}}\|_2}{\|\bar{x}_{\text{true}}\|_2}$ as well as the inner product of each recovered principal component with the true principal component.

This is done after centering the data with the recovered mean with the respective true principal component. We compare this for increasing number of observations $\mathbf{x}_i$ (up to 5000), and repeat this for 500 iterations to get the average inner product. Figure 4.5 shows the results.

Recall that the inner product of two vectors is near 1 when they are close to each other, and near zero when they are far apart. Thus, an inner product close to 1 implies that the reconstruction is close to the actual value.

We will use the term inner product in this section to denote the inner product of the reconstructed principal component with the actual principal component for brevity.

We can see that SDP performs better than RP in reconstructing the true mean with a lower relative error. Furthermore, SDP reconstructs the principal components with fewer observations compared to RP. The dotted lines represent our construction using SDP, while the solid lines represent the construction using RP. For example, with about 4000 observations, SDP could reconstruct the three principal components with high accuracy (dotted lines close to 1), but RP could only reconstruct the first two.

4.4 Our Experiments With Actual Data

We now conduct our experiments with two datasets: the colon dataset for a quick sanity check and the MNIST dataset. More information about these datasets can be found in Appendix A. We repeat all experiments for 100 simulations.

We first look at the colon dataset, which has $n = 62$ observations, and $p = 2000$ features. As SDP is dependent on the number of observations n, rather than the dimensions p or k, we want to see how well our algorithm constructs the principal components of the colon dataset.

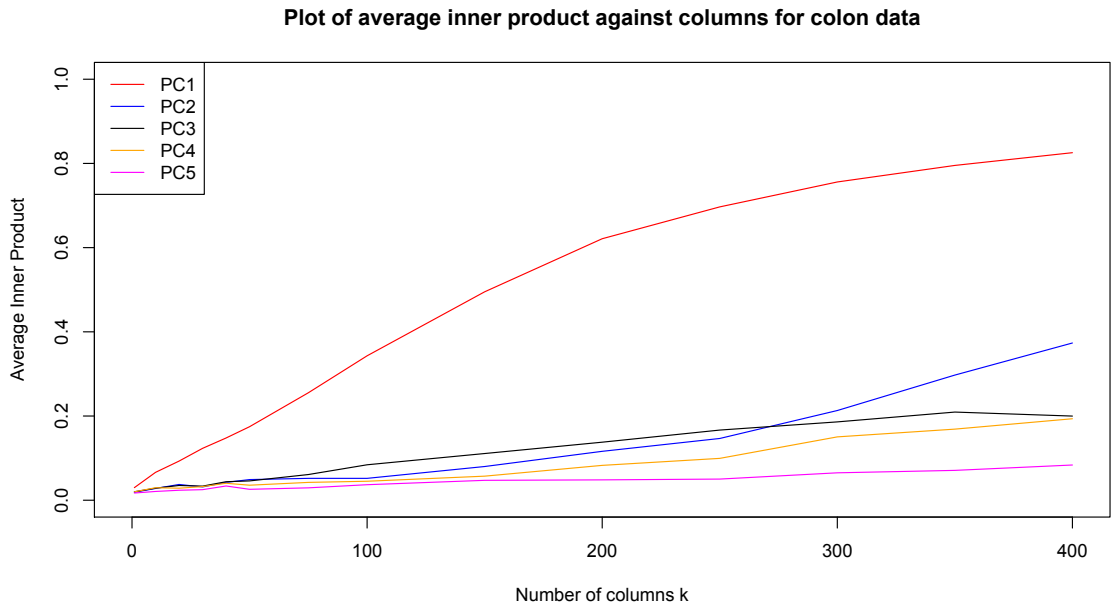

Figure 4.6: Inner Product Comparison For First Five PCs Of Colon Data

Figure 4.6 shows the average inner product of the first five PCs using SDP. As $n = 62$, we do not expect to reconstruct our principal components, but we do see that we can reconstruct some of the direction of the first principal component.

We then compare the reconstruction of the first principal component for all three methods (Qi, Farhad, and ours). Figure 4.7 shows the inner product for all three methods. We include error bounds, where the corresponding dotted lines are 3 standard deviations away from the mean.

With the colon dataset, there doesn't seem to be any difference between Qi's

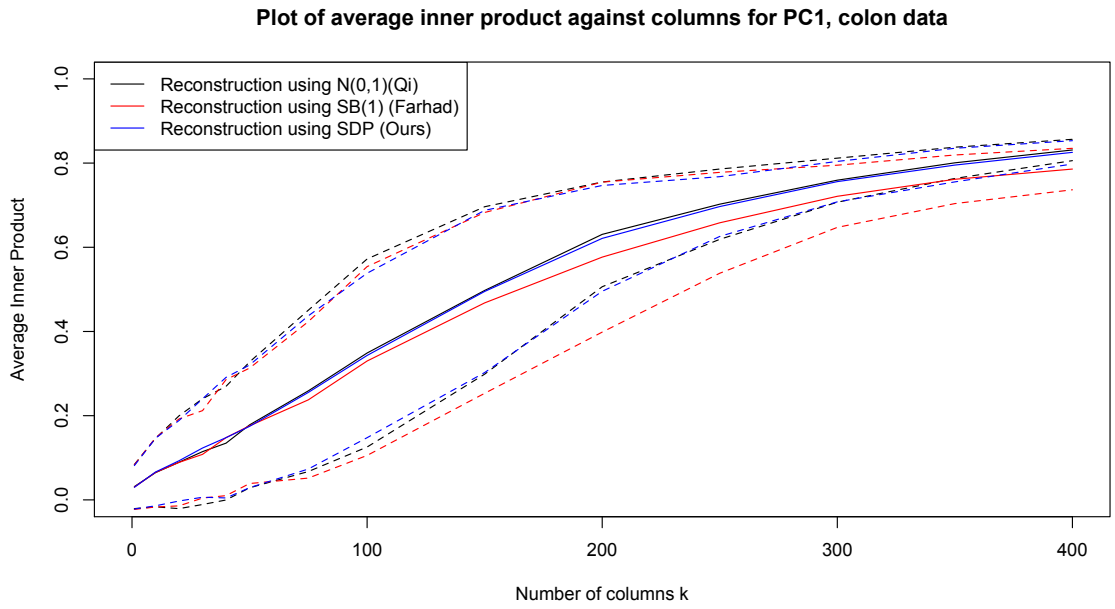

Figure 4.7: Comparison Of Inner Product For First PC For All Three Methods

method and our method for the reconstruction of the first principal component. Farhad's method using SB(1) seems to have a lower average inner product, with wider error bounds. However, we shouldn't read too much into this since n is relatively small.

We now look at the MNIST dataset, which has $n = 10,000$ observations, and $p = 784$ features.

First, we look at the number of principal components we can reconstruct with our method. Figure 4.8 shows our results. Generally, it seems that when $k > 75$, the average inner product converges to 1. This indicates that our method is able to reconstruct the principal components.

Second, we consider how accurate our reconstruction is compared with

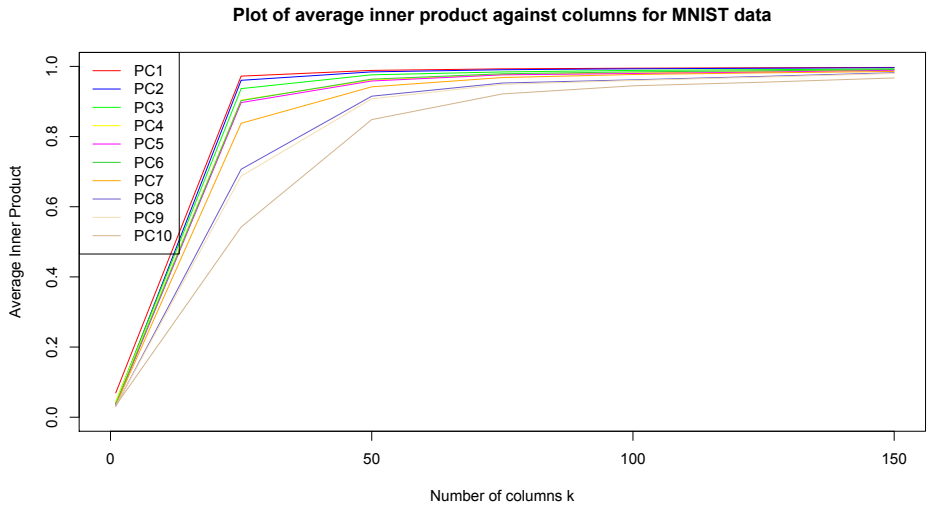

Figure 4.8: Inner Product Comparison For First Ten PCs Of MNIST Data

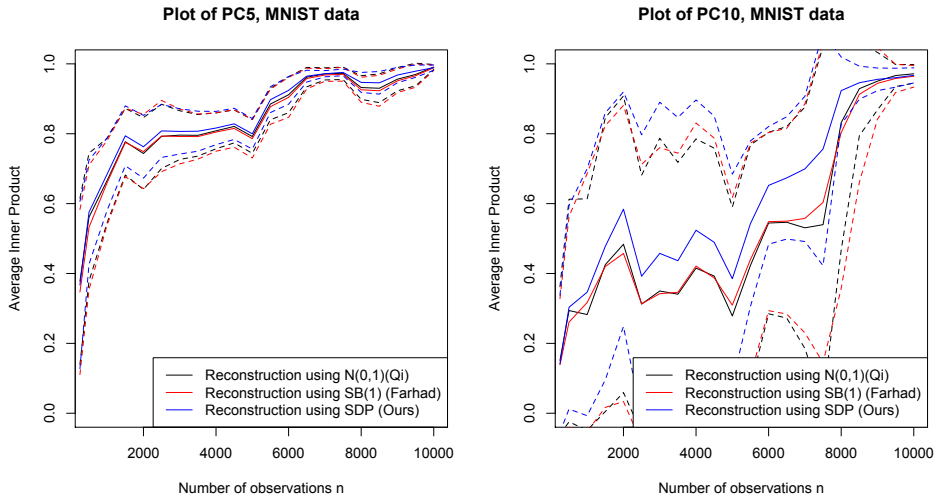

Figure 4.9: Comparison Of Inner Product For Fifth And Tenth PC For All Three Methods

other methods. We pick the 5th and the 10th PC to be reconstructed. We do not pick the earlier PCs, since all three methods reconstruct them well.

Figure 4.9 shows our results. Note that we use the same sample for all three

datasets, but generate random matrices differently. This causes the plots to be jagged.

We see almost no difference for the plot of PC5, even though our method performs marginally better than the other two methods. However, for the plot of PC10, we see that our method reconstructs more of the tenth PC with fewer observations. Furthermore, the error bounds are correspondingly higher than the error bounds of the other two methods.

In both synthetic and real data, our method has a noticeable effect in reconstructing the principal components. Thus, for a fixed n, our method generally allows us to reconstruct and estimate more principal components accurately than Farhad's method.

We omit the plots of Farhad's method using $SB(s)$, with $s > 1$. This is due to the fact that the variance in estimation of the covariance matrix increases as the parameter s increases. Thus, this affects the estimate of the eigenvectors. Since our method performs better than Farhad's method (using $s = 1$), then our method necessarily performs better than Farhad's method with parameters $s \geq 1$, which is also demonstrated by our empirical results.

4.5 Conclusion

Our construction of the projection matrix R performs extremely well and extends both Farhad's and Qi's methods [20, 22].

Our methods show that we can reconstruct more principal components accurately with the same number of observations compared to Farhad's and Qi's

method, when looking at the colon and MNIST dataset.

Furthermore, we reduce the cost of storage of R_is, as they can be encoded as a p dimensional vector of signs, rather than a $p \times k$ matrix of 1s and 0s.

We believe that our work could lead to similar future random projection algorithms which require an accurate reconstruction of the covariance matrix by creating random projections for individual observations, such as Canonical Correlation Analysis for example.

Overall, we hope that our structure of random projections opens up the door to faster and more accurate random projection algorithms for matrix factorizations.

CHAPTER 5

RANDOM PROJECTIONS WITH CONTROL VARIATES

In previous chapters, we explored how changing the structure of the random projection matrix may provide speedups and sometimes a more accurate estimate of parameters required. In this chapter, we will show how borrowing an idea from Monte Carlo integration and storing some other information of the data matrix X can lead to more accurate estimates for all random projection matrix structures.

The previous chapters showed that the methods used in construction and application of the random projection matrix R to the vectors $\mathbf{x}_i$s have tradeoffs. Very sparse random projections, FJLT, and the SRHT are fast methods with a tradeoff in accuracy. The former uses extremely sparse R for quick matrix multiplication (optimal R has about $\frac{\sqrt{p}-1}{\sqrt{p}}$ zero entries), and the latter two uses the recursive property of the Hadamard matrix for quick matrix vector multiplication. Dense R with entries generated from the Normal or the Rademacher distribution gives more accurate estimates and desparsifies data but at a cost of speed.

We will propose a method *Random Projections with Control Variates* (RPCV), which is used in conjunction with the above types of different random projection matrices. Thus, if a particular type of random projection matrix is chosen by the user based on the data X, RPCV can be used to get a variance reduction in the estimation of Euclidean distances and inner products between pairs of vectors $\mathbf{x}_i, \mathbf{x}_j$ with a negligible extra cost in speed and storage space. These measures of distance can then be used for in clustering, classification [18], and set

resemblance problems [11].

5.1 Notation

We denote $R \in \mathbb{R}^{p \times k}$ to be a random projection matrix. We let $X \in \mathbb{R}^{n \times p}$ to be our data matrix, where each row $\mathbf{x}_i^T \in \mathbb{R}^p$ is a p dimensional observation. The random projection equation is then given by:

$$V = XR \tag{5.1}$$

Consider the random matrix R written as

$$R = [\mathbf{r}_1 \mid \mathbf{r}_2 \mid \ldots \mid \mathbf{r}_k] \tag{5.2}$$

where each $\mathbf{r}_i$ is a column vector with i.i.d. entries. Then for a *fixed* row $\mathbf{x}_i^T$, we have that for all j, $v_{ij} = \mathbf{x}_i^T \mathbf{r}_j$ is a random variable from the same distribution. Here, we focus on each v_{ij} as a single element, rather than seeing $v_{i1}, \ldots, v_{ik}$ comprising the row vector $\mathbf{v}_{i.}^T$.

5.2 Control Variates

Given the notion of each v_{ij} as a random variable, we introduce control variates. Control variates are a technique in Monte Carlo simulation using random variables for variance reduction. A more thorough explanation found in Ross [23].

The method of control variates assumes we use the same random inputs to estimate $\mathbb{E}[A] = \mu_A$, for which we know B with $\mathbb{E}[B] = \mu_B$. We call B our control

variate. Then to estimate $\mathbb{E}[A] = \mu_A$ from some distribution A, we can instead compute the expectation of

$$\mathbb{E}[A + c(B - \mu_B)] \quad = \quad \mathbb{E}[A] + c\mathbb{E}[B - \mu_B] \quad = \quad \mu_A \tag{5.3}$$

which is an unbiased estimator of μ_A for some constant c. This value of c which minimizes the variance is given by

$$\hat{c} = -\frac{\text{Cov}(A, B)}{\text{Var}(B)} \tag{5.4}$$

and thus we write

$$\text{Var}[A + c(B - \mu_B)] = \text{Var}(A) - \frac{(\text{Cov}(A, B))^2}{\text{Var}(B)} \tag{5.5}$$

In our random projection scenario for a fixed i, we can think of a random variable A as some v_{ij}, where

$$\mathbb{E}[v_{i.}] = \mathbb{E}\left[\frac{1}{k}\sum_{m=1}^{k} v_{im}\right] = \frac{1}{k}\sum_{m=1}^{k}\left(\sum_{n=1}^{p} x_{i,n} r_{nm}\right) \tag{5.6}$$

under the law of large numbers.

Intuitively, we then need to find some distribution B where the variables b_i are correlated with v_{ij} to get good variance reduction. To do this, B necessarily needs to fulfill two conditions.

Condition 1: Since each realization v_{ij} is the sum of p random variables $r_{1j}, r_{2j}, \ldots, r_{pj}$, we need to have y_i constructed from these same random variables *and* also correlated with each $x_{i1}, \ldots, x_{ip}$ in order to get a variance reduction.

Condition 2: We need to know the actual value of μ_B, the mean of B.

This seems like a chicken and egg problem since any μ_B that is related to both $x_{i.}$, $r_{.j}$ would be of some form of either the Euclidean distance or the inner product, both of which we want to estimate in the first place.

We solve this problem by considering an expression that relates both the Euclidean distance and the inner product simultaneously.

5.3 Related Work

We draw inspiration from the works of Li and Church [10] [11] [12]. In these papers, marginal information such as margin counts or margin norms from data is pre-computed and stored. This extra information is then used with asymptotic maximum likelihood estimators to estimate parameters of interests.

We also store marginal information from our matrix X, but instead use this information to determine a control variate, rather than a maximum likelihood estimator. We compute and store all the n norms $\|\mathbf{x_i}\|^2$ from our X. Computing all these norms are cheap as they are of order $O(np)$, and can be done when reading in the data at the same time.

Furthermore if the data is normalized (normalizing is also of order $O(np)$ which we usually take for granted), we get the norms $\|\mathbf{x}_i\|_2^2 = 1$ for free.

5.3.1 Random Projections With Marginal Information

We will give a brief description of the algorithm used in Li's paper[11] as we use it for one of our baselines. Li's paper focused on estimating the inner product between any two vectors $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$, conditioning on knowing the value of the norms $\|\mathbf{x}_i\|_2^2, \|\mathbf{x}_j\|_2^2$. Li used a random projection matrix R with $N(0, 1)$ entries, but his method can be extended for other R. We briefly step through the key ideas

of his paper.

Given the matrix V in Equation 5.1, consider an arbitrary tuple (v_{is}, v_{js}), read as the s^{th} entry in the row $\mathbf{v}_i$, and the s^{th} entry in the row $\mathbf{v}_j$. Each tuple has the following bivariate normal distribution:

$$\begin{pmatrix} v_{ik} \\ v_{jk} \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} m_i & a \\ a & m_j \end{pmatrix}\right) \tag{5.7}$$

where $m_i = \|\mathbf{x}_i\|_2^2$, $m_j = \|\mathbf{x}_j\|_2^2$, and a the inner product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$.

As the bivariate normal pdf is given by:

$$f(\mathbf{x}) = \frac{1}{2\pi \, |\Sigma|^{-\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)\right\} \tag{5.8}$$

then the likelihood function of k such observations is given by:

$$L(a \mid \mathbf{v}) = \prod_{i=1}^{k} \frac{1}{2\pi \, |\Sigma|^{-\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{v}-\mu)^T \Sigma^{-1} (\mathbf{v}-\mu)\right\}$$

where μ, Σ is given in Equation 5.7.

Since we know the norms m_i, m_j, then it suffices to find a which maximizes the likelihood (or log-likelihood).

Li gives the log-likelihood function to be:

$$l(a) = -\frac{k}{2}\log(m_1 m_2 - a^2) - \frac{k}{2}\frac{1}{m_1 m_2 - a^2}\sum_{s=1}^{k}(v_{is}^2 m_j - 2v_{is}v_{js}a + v_{js}^2 m_i) \tag{5.9}$$

and the value of a which maximizes this can be found via root optimization methods, such as Newton Raphson.

Therefore, we can view Li's method working for normal R as the following algorithm in three steps:

1. Find the distribution $p(v)$ of the tuple (v_{is}, v_{js}).

2. Find the likelihood $L(a)$ of the k observations $(v_{is}, v_{js})_{\{s=1\}}^{k}$ where a is the actual inner product.

3. Find the value of a which maximizes this likelihood.

For other R, Li showed that the estimates $v_{i.}$ converge to a scaled $N(0,1)$, thus results hold asymptotically.

5.4 Our Contributions: Random Projections With Control Variates

We will describe and illustrate the process of random projections with control variates in this section.

Without loss of generality, suppose we had $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^p$. Consider $\mathbf{v}$ given by $X\mathbf{r}$. As an illustrative example in the case where $p = 2$, we would have:

$$V = \begin{pmatrix} v_1 \\ \\ v_2 \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} \\ \\ x_{21} & x_{22} \end{pmatrix} \begin{pmatrix} r_1 \\ \\ r_2 \end{pmatrix} = X\mathbf{r} \tag{5.10}$$

for one column of R. We do matrix multiplication $X\mathbf{r}$ and get v_1, v_2.

In the next two sections, we will give the control variate to estimate the Euclidean distance and the inner product. We will also give the respective optimal control variate correction c, and the respective first and second moments of the expression $A + c(B - \mu_B)$. This allows us to compute a more accurate estimate for the Euclidean distance and the inner product, as well as place probability bounds on the errors of our estimates.

5.4.1 RPCV For Euclidean Distance

Suppose we computed V as above. The following theorem shows us how to estimate the Euclidean distance with our control variate.

Theorem 5.4.1. Let one realization of $A = (v_1 - v_2)^2$, which is our Euclidean distance in expectation. Let one realization of B to be $(v_1 - v_2)^2 + 2v_1v_2 = v_1^2 + v_2^2$ with mean $\mu_B = \|\mathbf{x}_1\|^2 + \|\mathbf{x}_2\|_2^2$. The Euclidean distance (in expectation) between these two vectors is given by $\mathbb{E}[A + c(B - \mu_B)]$, and we can compute $c := -\text{Cov}(A, B)/\text{Var}(B)$ from our matrix V directly, using the empirical covariance $\text{Cov}(A, B)$ and empirical variance $\text{Var}(B)$.

Proof. We have

$$\begin{aligned}
&\mathbb{E}[(v_1 - v_2)^2] + 2\mathbb{E}[v_1v_2] \\
&= \|\mathbf{x}_1 - \mathbf{x}_2\|^2 + 2\langle \mathbf{x}_1\mathbf{x}_2 \rangle && (5.11) \\
&= \|\mathbf{x}_1\|^2 + \|\mathbf{x}_2\|^2 - 2\langle \mathbf{x}_1, \mathbf{x}_2 \rangle + 2\langle \mathbf{x}_1, \mathbf{x}_2 \rangle && (5.12) \\
&= \|\mathbf{x}_1\|^2 + \|\mathbf{x}_2\|^2 && (5.13)
\end{aligned}$$

□

We derive the following lemma to help us compute the first and second moments required.

Lemma 5.4.1. Suppose we assume that our matrix R has i.i.d. entries, where each r_{ij} has mean $\mu = 0$, second moment $\mu_2 = 1$, and fourth moment μ_4. Then

under this set up for Euclidean distances in Theorem 5.4.1, we have

$$\begin{aligned}\mathbb{E}[A^2] = \quad & \mu_4 \sum_{j=1}^{p} (x_{1j} - x_{2j})^4 \\ & + 6 \sum_{u=1}^{p-1} \sum_{v=u+1}^{p} (x_{1u} - x_{2u})^2 (x_{1v} - x_{2v})^2 \end{aligned} \tag{5.14}$$

$$\begin{aligned}\mathbb{E}[B^2] = \mu_4 & \sum_{j=1}^{p} (x_{1j}^4 + x_{2j}^4) + 6 \sum_{u=1}^{p-1} \sum_{v=u+1}^{p} (x_{1u}^2 x_{1v}^2 + x_{2u}^2 x_{2v}^2) \\ & + 4 \sum_{u=1}^{p-1} \sum_{v=u+1}^{p} (x_{1u} x_{1v} x_{2u} x_{2v}) + \mu_4 \sum_{j=1}^{p} x_{1j}^2 x_{2j}^2 \\ & + \sum_{i \neq j}^{p} x_{1i}^2 y_{2j}^p \end{aligned} \tag{5.15}$$

$$\begin{aligned}\mathbb{E}[AB] = 4 & \sum_{u=1}^{p-1} \sum_{v=u+1}^{p} (x_{1u} - x_{2u}) (x_{1v} - x_{2v}) (x_{1u} x_{1v} + x_{2u} x_{2v}) \\ & + \mu_4 \sum_{j=1}^{p} (x_{1j} - x_{2j})^2 (x_{1j}^2 + x_{2j}^2) \\ & + \sum_{i \neq j} (x_{1i} - x_{2i})^2 (x_{1i}^2 + x_{2j}^2) \end{aligned} \tag{5.16}$$

Proof. We repeatedly apply Lemma B.0.1 in the Appendix. □

Thus, by following Lemma 5.4.1, we are able to derive expressions for the optimal control variate correction c in our procedure as follows.

Theorem 5.4.2. The optimal value c is given by:

$$c = -\frac{\text{Cov}(A, B)}{\text{Var}[B]} \tag{5.17}$$

where we have:

$$\text{Cov}(A, B) = \mathbb{E}[AB - A\mu_B - B\mu_A + \mu_A \mu_B] \tag{5.18}$$

and:

$$\text{Var}[B] = \mathbb{E}[B^2] - (\mathbb{E}[B])^2 \tag{5.19}$$

They expand to:

$$\begin{aligned}\text{Cov}(A, B) = 4 \sum_{u=1}^{p-1} \sum_{v=u+1}^{p} (x_{1u} - x_{2u})(x_{1v} - x_{2v})(x_{1u}x_{1v} + x_{2u}x_{2v}) \\ + (\mu_4 - 1) \sum_{j=1}^{p} (x_{1j} - x_{2j})^2 (x_{1j}^2 + x_{2j}^2)\end{aligned} \tag{5.20}$$

and:

$$\begin{aligned}\text{Var}[B] = (\mu_4 - 1) \sum_{j=1}^{p} (x_{1j}^4 + x_{2j}^4) + 4 \sum_{u=1}^{p-1} \sum_{v=u+1}^{p} (x_{1u}^2 x_{1v}^2 \\ + x_{2u}^2 x_{2v}^2) + 4 \sum_{u=1}^{p-1} \sum_{v=u+1}^{p} x_{1u}x_{1v}x_{2u}x_{2v} \\ + (\mu_4 - 2) \sum_{j=1}^{p} x_{1j}^2 x_{2j}^2 - \sum_{i \neq j} x_{1i}^2 x_{2j}^2\end{aligned} \tag{5.21}$$

In practice, we compute c empirically from the random projection estimates. We are also able to derive the first and second moments of $A + c(B - \mu_B)$ for Euclidean distances.

Theorem 5.4.3. The first and second moments are:

$$\mathbb{E}[A + c(B - \mu_B)] = \mathbb{E}[A] + c\mathbb{E}[B - \mu_B] = \mathbb{E}[A] \tag{5.22}$$

and:

$$\mathbb{E}[(A + c(B - \mu_B))^2 = \mathbb{E}[A^2 + 2cAB - 2c\mu_B A + c^2 B^2 - 2c^2 \mu_B B + c^2 \mu_B^2] \tag{5.23}$$

where we substitute in the values of $\mathbb{E}[A^2], \mathbb{E}[AB], \mathbb{E}[B^2]$ from Lemma 5.4.1.

5.4.2 RPCV For Inner Product

Suppose we computed V as above. The following theorem shows us how to estimate the inner product with our control variate.

Theorem 5.4.4. Let one realization of $A = v_1 v_2$, which is our inner product in expectation. Let one realization of B to be $(v_1 - v_2)^2 + 2v_1 v_2 = v_1^2 + v_2^2$ with mean $\mu_B = \|\mathbf{x}_1\|^2 + \|\mathbf{x}_2\|_2^2$. The inner product between these two vectors is given by $\mathbb{E}[A + c(B - \mu_B)]$, and we can compute $c := -\text{Cov}(A, B)/\text{Var}(B)$ from our matrix V directly, using the empirical covariance $\text{Cov}(A, B)$ and empirical variance $\text{Var}(B)$.

The optimal control variate c in this procedure is given by the next theorem.

Theorem 5.4.5. The optimal value of c is given by:

$$c = -\frac{\text{Cov}(A, B)}{\text{Var}[B]} \tag{5.24}$$

where:

$$\begin{aligned} \text{Cov}(A, B) &= \mathbb{E}[AB - A\mu_B - B\mu_A + \mu_A\mu_B] \\ &= (\mu_4 - 1) \sum_{j=1}^{p} x_{1j} x_{2j} (x_{1j}^2 + x_{2j}^2) \\ &\quad + \sum_{i \neq j} x_{1i} x_{2j} (x_{1i} x_{1j} + x_{2i} x_{2j}) \end{aligned} \tag{5.25}$$

and the value of $\text{Var}[B]$ taken from the result in Theorem 5.4.2.

5.4.3 The optimal control variate correction c

While we have computed an expression c in terms of the first and second moments of our distributions, they are not at all intuitive from first sight. Therefore,

we consider what the optimal value of c would be if the random matrix R had i.i.d. entries $r_{ij} \sim N(0,1)$. Thus, we take a second look at the bivariate normal distribution in Equation 5.7.

Theorem 5.4.6. *For $r_{ij} \sim N(0,1)$, and $V = XR$, the optimal control variate correction c_{ED} for the Euclidean distance is given by*

$$c_{ED} = -\frac{(m_1 - a)^2 + (m_2 - a)^2}{(m_1^2 + m_2^2 + 2a^2)} \tag{5.26}$$

Proof. We can write the control variate correction c for the Euclidean distance as

$$c_{\text{ED}} = -\frac{\text{Cov}(v_1^2 + v_2^2, v_1^2 + v_2^2 - 2v_1v_2)}{\text{Var}\left(v_1^2 + v_2^2\right)} \tag{5.27}$$

$$= -\frac{\text{Cov}(\mathbf{v}_1^T\mathbf{v}_2, \mathbf{v}_1{}^T H\mathbf{v}_2)}{\text{Var}\left(\mathbf{v}_1^T\mathbf{v}_2\right)} \tag{5.28}$$

where $H = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$. Next, expanding the numerator gives

$$\text{Cov}(\mathbf{v}_1^T\mathbf{v}_2, \mathbf{v}_1{}^T H\mathbf{v}_2) = \mathbb{E}[\mathbf{v}_1^T\mathbf{v}_2\mathbf{v}_1^T H\mathbf{v}_2] - \mathbb{E}\left[\mathbf{v}_1^T\mathbf{v}_2\right]\mathbb{E}\left[\mathbf{v}_1^T H\mathbf{v}_2\right] \tag{5.29}$$

For $(v_1, v_2) \sim N(0, \Sigma)$, we have the identities

$$\mathbb{E}\left[\mathbf{v}_1^T\mathbf{v}_2\mathbf{v}_1^T H\mathbf{v}_2\right] = \text{Tr}(\Sigma(H + H^T)\Sigma) + \text{Tr}(\Sigma)\text{Tr}(H\Sigma) \tag{5.30}$$

$$= 2(m_1 - a)^2 + 2(m_2 - a)^2 + (m_1 + m_2)(m_1 + m_2 - 2a) \tag{5.31}$$

$$\mathbb{E}[\mathbf{v}_1^T\mathbf{v}_2] = m_1 + m_2 \tag{5.32}$$

$$\mathbb{E}[\mathbf{v}_1^T H\mathbf{v}_2] = m_1 + m_2 - 2a \tag{5.33}$$

and therefore, we have

$$\text{Cov}(\mathbf{v}_1^T\mathbf{v}_2, \mathbf{v}_1{}^T H\mathbf{v}_2) = 2(m_1 - a)^2 + 2(m_2 - a)^2 \tag{5.34}$$

The denominator expands to be

$$\text{Var}\left(\mathbf{v}_1^T\mathbf{v}_2\right) = \text{Tr}\left(\Sigma(2I)\Sigma\right) \tag{5.35}$$

$$= 2(m_1^2 + m_2^2 + 2a^2) \tag{5.36}$$

Simplifying, we get:

$$c_{\text{ED}} = -\frac{(m_1 - a)^2 + (m_2 - a)^2}{(m_1^2 + m_2^2 + 2a^2)} \tag{5.37}$$

□

Theorem 5.4.7. *For $r_{ij} \sim N(0,1)$, and $V = XR$, the optimal control variate correction c_{IP} for the inner product is given by*

$$c_{IP} = -\frac{m_1 a + m_2 a}{m_1^2 + m_2^2 + 2a^2} \tag{5.38}$$

Proof. Analogous to the proof of Theorem 5.4.6, we express

$$\text{Cov}(v_1^2 + v_2^2, v_1 v_2) = \text{Cov}\left(\mathbf{v}_1^2\mathbf{v}_2, \mathbf{v}_1^T H\mathbf{v}_2\right) \tag{5.39}$$

where $H = \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. Therefore, we similarly compute

$$\mathbb{E}\left[\mathbf{v}_1^T\mathbf{v}_2\mathbf{v}_1^T H\mathbf{v}_2\right] = \text{Tr}(\Sigma(H + H^T)\Sigma) + \text{Tr}(\Sigma)\text{Tr}(H\Sigma) \tag{5.40}$$

$$= 2m_1 a + 2m_2 a + (m_1 + m_2)(a) \tag{5.41}$$

$$\mathbb{E}[\mathbf{v}_1^T\mathbf{v}_2] = m_1 + m_2 \tag{5.42}$$

$$\mathbb{E}[\mathbf{v}_1^T H\mathbf{v}_2] = a \tag{5.43}$$

which results in

$$c_{\text{IP}} = \frac{m_1 a + m_2 a}{m_1^2 + m_2^2 + 2a^2} \tag{5.44}$$

□

Without loss of generality, we assume that our data is normalized such that $m_1 = m_2 = 1$. In this case, we can compute the variance reduction for our Euclidean distances and inner products respectively.

Theorem 5.4.8. *Given $r_{ij} \sim N(0,1)$ and $V = XR$, then for any pair $\mathbf{x}_i, \mathbf{x}_j$*

1. *The variance of the estimate of the Euclidean distance between the pair is given by*

$$\sigma_{ED} = 8(1-a)^2 \tag{5.45}$$

2. *The variance of the estimate of the Euclidean distance with the control variate correction between the pair is given by*

$$\sigma_{EDCV} = 8(1-a)^2 - \frac{4(1-a)^4}{(1+a^2)} \tag{5.46}$$

3. *The variance of the estimate of the inner product between the pair is given by*

$$\sigma_{IP} = 1 + a^2 \tag{5.47}$$

4. *The variance of the estimate of the inner product with the control variate correction between the pair is given by*

$$\sigma_{IPCV} = 1 + a^2 - \frac{4a^2}{1+a^2} \tag{5.48}$$

Proof. This follows from direct substitution of the optimal control variate corrections in Theorems 5.4.6 and 5.4.7 into Equation 5.5. □

Theorem 5.4.8 allows us to analyze the effect of our control variate correction, given in Figure 5.1.

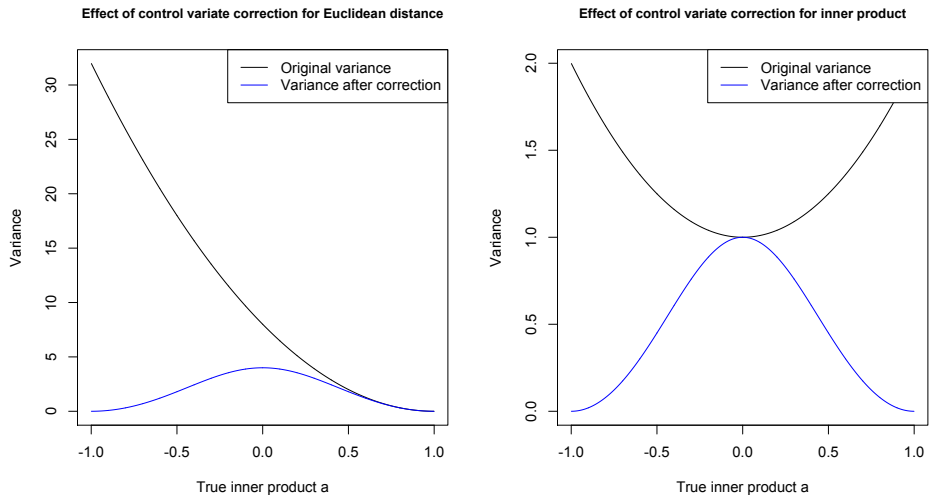

Figure 5.1: Effects of control variate correction on estimates

Recall that when observations are normalized, we have the Euclidean distance between any pair of vectors being in the range $[0, 2]$, and the inner product between any pair of vectors being in the range $[0, 1]$.

We can now analyze the effect of our control variate correction on Euclidean distance. For vectors $\mathbf{x}_i, \mathbf{x}_j$ in the same direction (inner product close to 1), we do not get much variance reduction in our estimate of Euclidean distance. Conversely, if the vectors were in opposite directions, then we would get a reasonable variance reduction in the estimates of their Euclidean distance.

Similarly, we analyze the effect of our control variate correction on the inner product. If the vectors $\mathbf{x}_i, \mathbf{x}_j$ are orthogonal to each other (inner product near 0), then we do not get much variance reduction from our control variate correction. Conversely, if the vectors share the same or opposite directions, then we would get a reasonable variance reduction in the estimates of their inner product.

Theorems 5.4.6, 5.4.7, and Li et al's result that the tuple $((v_{is}, v_{js})$ converges

to a bivariate normal even if r_{ij} from R do not come from $N(0,1)$ suggests an alternative method of computing the control variate correction.

Instead of computing the control variate correction c_{ED} empirically from our data, we could choose to either compute the vanilla estimate $\hat{a} = v_{is}v_{js}$ for the inner product, or $\hat{a}$ using Li's method. We can then substitute $\hat{a}$ into the results of Theorem 5.4.6 to compute the optimal control variate correction, using the fact that we get convergence to a bivariate normal when the number of observations increase. In fact, since we are storing the marginal norms, we might as well compute $\hat{a}$ via Li's method, and use this to compute c_{ED} directly.

Similarly, we could compute the control variate correction c_{IP}, but only using the ordinary estimate $\hat{a} = v_{is}v_{js}$ for the inner product.

5.4.4 Motivation For Computing First And Second Moments

In Chapter 2, we gave a strategy (Strategy 2.4.1) on how to compute probability bounds, and had:

$$\mathbb{P}\left[(1-\epsilon)\|\mathbf{x}\|_2^2 \leq \frac{S_k^{\text{norm}}}{k} \leq (1+\epsilon)\|\mathbf{x}\|_2^2\right] \leq 1 - f_1(\epsilon,k_1) - f_2(\epsilon,k_2) \tag{5.49}$$

$$\mathbb{P}\left[(1-\epsilon)\|\mathbf{x}_i-\mathbf{x}_j\|_2^2 \leq \frac{S_k^{\text{ED}}}{k} \leq (1+\epsilon)\|\mathbf{x}_i-\mathbf{x}_j\|_2^2\right] \leq 1 - f_1(\epsilon,k_1) - f_2(\epsilon,k_2) \tag{5.50}$$

$$\mathbb{P}\left[(1-\epsilon)\langle\mathbf{x}_i,\mathbf{x}_j\rangle \leq \frac{S_k^{\text{IP}}}{k} \leq (1+\epsilon)\langle\mathbf{x}_i,\mathbf{x}_j\rangle_2^2\right] \leq 1 - 2f_1(\epsilon,k_1) - 2f_2(\epsilon,k_2) \tag{5.51}$$

If we construct R with i.i.d. $r_{ij} \sim N(0,1)$, or $r_{ij} \sim \{\pm 1\}$, then we can easily find similar probability bounds for the Euclidean distance by setting $\|\mathbf{v}\| = \|\mathbf{v}_1 - \mathbf{v}_2\|$. In the RPCV case, each element $v_{1i} - v_{2i}$ in $\|\mathbf{v}_1 - \mathbf{v}_2\|$ now corresponds to:

$$(v_{1i}-v_{2i})^2 + c(\mathbf{v}_{1i}^2 + \mathbf{v}_{2i}^2 - \|\mathbf{x}\|^2 - \|\mathbf{x}_2\|^2) \tag{5.52}$$

and thus we need to find probability bounds for this expression.

In these cases, we had S_k to be the sum of our estimates in the form of v, and can be seen as a realization of A. Using random projections with control variates, our S_k now has an additional term B, which is given by $c(v_i^2 + v_j^2 - \|\mathbf{x}_i\|_2^2 - \|\mathbf{x}_j\|_2^2)$.

Getting bounds $f_(\epsilon, k_i)$ in Strategy 2.4.1 involved getting the second moments of S_k. Thus, we also had to compute the first and second moments for the expression $A + c(B - \mu_B)$ for $A = (v_1 - v_2)^2$, $B = \|\mathbf{v}_1\|_2^2 + \|\mathbf{v}_2\|_2^2$ for both the Euclidean distances and inner product. This requires pre-computing $\hat{c}$ for the Euclidean distance, and $\tilde{c}$ for the inner product so we can substitute them into the value of the first and second moments.

For R constructed with r_{ij} from other distributions, computing these bounds are a bit more involved.

5.4.5 Overall Computational Time

We need to compute the empirical covariance between all pairs A and B as well as the variance of B, which takes an additional $O(k)$ time. Since the vectors we need to compute this covariance are the elements of V, we do not need to do further computation to get them. Furthermore, computing the covariance takes the same order of time as finding the Euclidean distance (or inner product) between the vectors $\mathbf{v}_i$, $\mathbf{v}_j$.

If we want a more accurate estimate of the inner product using Li's method, we can either use a root finding method to find a where $f(a) = 0$, or use the cubic formula to get the root(s) of a degree 3 polynomial. The time for these methods

Table 5.1: Random Projection Matrices For RPCV

R	Type
R_1	Entries i.i.d. from $N(0,1)$
R_2	Entries i.i.d. from $\{-1,1\}$ with equal probability
R_3	Entries i.i.d. from $\{-\sqrt{p}, 0, \sqrt{p}\}$ with probabilities $\{\frac{1}{2p}, 1-\frac{1}{p}, \frac{1}{2p}\}$ for $p=5$
R_4	Entries i.i.d. from $\{-\sqrt{p}, 0, \sqrt{p}\}$ with probabilities $\{\frac{1}{2p}, 1-\frac{1}{p}, \frac{1}{2p}\}$ for $p=10$
R_5	Constructed using the Subsampled Randomized Hadamard Transform (SRHT)

are are bounded above by some constant number of operations.

5.5 Our Experiments

In our experiments, we use five different types of random projection matrices as shown in Table 5.1. We pick these five types of random projection matrices as they are commonly used random projection matrices.

We use $N(0,1)$ to denote the Normal distribution with mean $\mu = 0$ and $\sigma^2 = 1$. We denote $(\mathbf{1})_p$ to be the length p vector with all entries being 1, and $(\mathbf{0})_p$ to be the length p vector with all entries being 0. We denote the baseline estimates to be the respective estimates given by using the type of random projection matrix R_i.

We run our simulations for 10000 iterations for every experiment, and use R with varying columns k ranging from 2 to 100.

Table 5.2: Generated Data $\mathbf{x}_1$, $\mathbf{x}_2$.

Pairs	$\mathbf{x}_1$	$\mathbf{x}_2$
Pair 1	Entries i.i.d. from $N(0,1)$	Entries i.i.d. from $N(0,1)$
Pair 2	Entries i.i.d. from standard Cauchy	Entries i.i.d. from standard Cauchy
Pair 3	Entries i.i.d. from Bernoulli(0.05)	Entries i.i.d. from Bernoulli(0.05)
Pair 4	Vector $[(\mathbf{1})_{p/2}, (\mathbf{0})_{p/2}]$	Vector $[(\mathbf{0})_{p/2}, (\mathbf{1})_{p/2}]$

5.5.1 Experiments With Synthetic Data

We first perform our experiments on a wide range of synthetic data. We look at normalized pairs of vectors $\mathbf{x}_1,\ \mathbf{x}_2 \in \mathbb{R}^{5000}$ generated from the following distributions in Table 5.2. In short, we look at data that can be Normal, heavy tailed (Cauchy), sparse (Bernoulli), and an adversarial scenario where the inner product is zero.

We first compare the relative bias of the estimates of Euclidean distance using RPCV against the baseline estimates for each projection R_i for a sanity check. Plots can be seen in Figure 5.2, and the relative bias goes to zero as expected.

We now look at the plots of the ratio ρ defined by

$$\rho = \frac{\text{Variance using control variate with } R_i}{\text{Variance using baseline with } R_i} \tag{5.53}$$

In Figure 5.3 for the Euclidean distance. ρ is a measure of the reduction in variance using RPCV with the matrix R_i rather than just using R_i alone. For this ratio, a fraction less than 1 means RPCV performs better than the baseline.

For all pairs $\mathbf{x}_i, \mathbf{x}_j$ except Cauchy, the reduction of variance of the estimates of the Euclidean distance using different R_is with RPCV converge quickly to

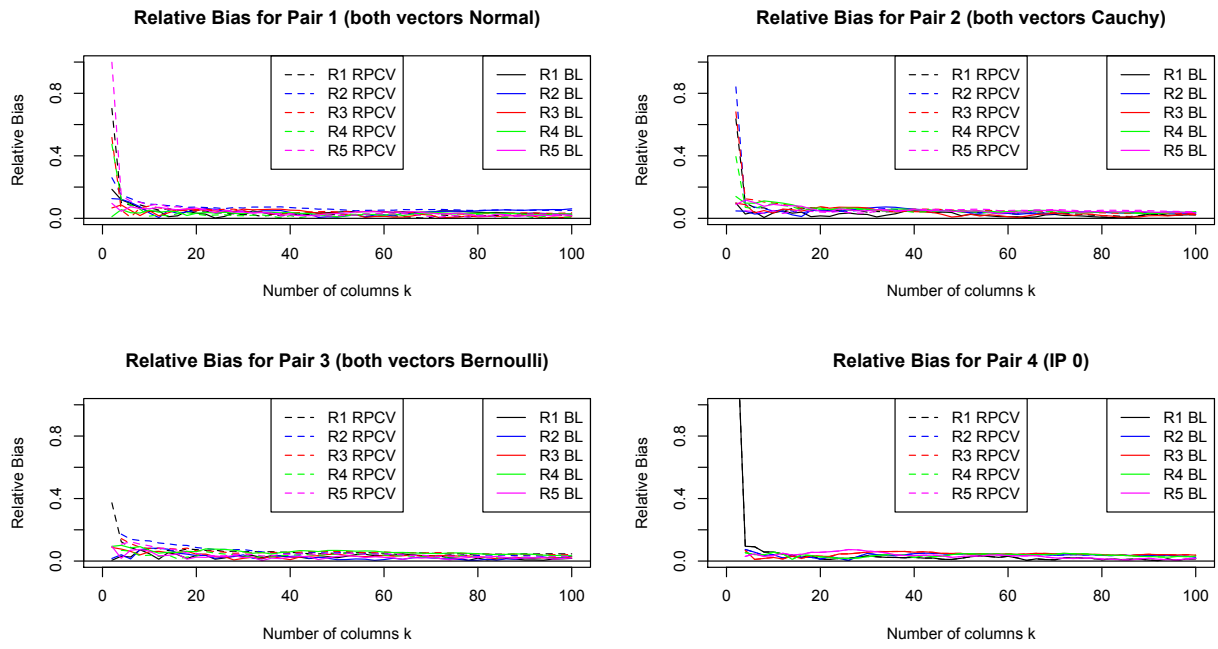

Figure 5.2: Plots Of Relative Bias In Synthetic Data

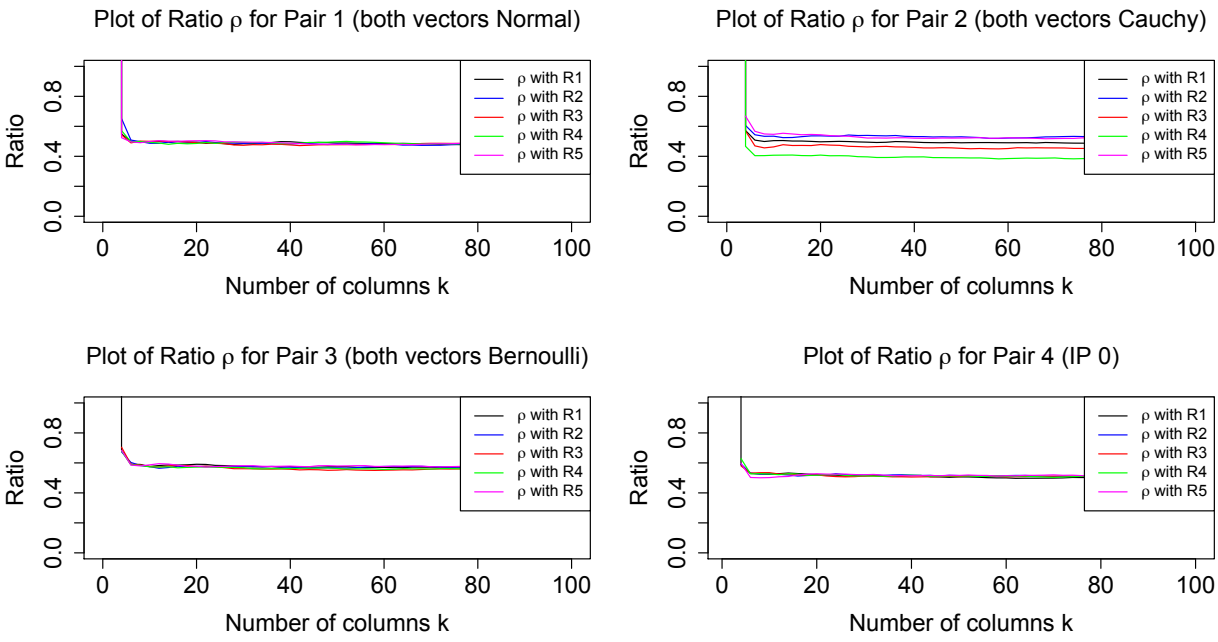

Figure 5.3: Plots Of ρ For Euclidean Distances For Synthetic Data

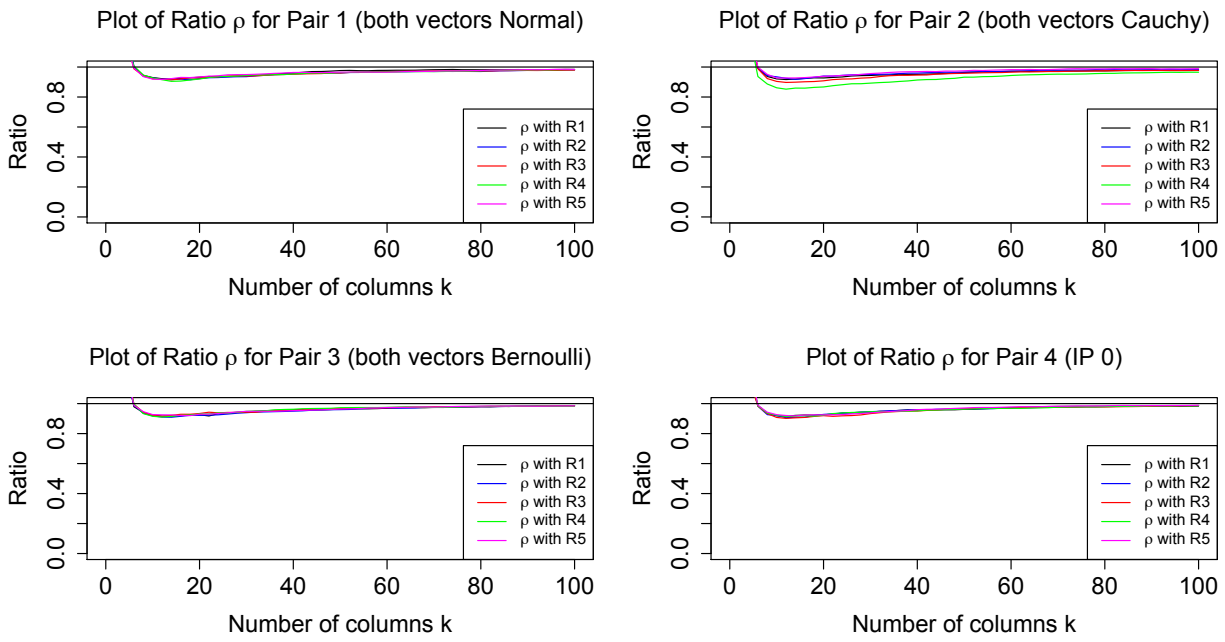

Figure 5.4: Plots Of ρ For Inner Product For Synthetic Data

around the same ratio. However, when data is heavy tailed, the choice of random projection matrix R_i with RPCV affects the reduction of variance in the estimates of the Euclidean distance, and sparse matrices R_i have a greater variance reduction for the estimates of the Euclidean distance.

We next look at the estimates of the inner product. In our experiments, we use Li's method as the baseline for computing the estimates of the inner product. Our rationale for doing this is that both Li's method and our method stores the marginal norms of X, thus we should compare our method with Li's method for a fair comparison. The ratio of variance reduction is shown in Figure 5.4.

As the number of columns k of the random projection matrix R increases, the variance reduction in our estimate of the inner product decreases, but then increases again up to a ratio just below 1. Since Li's method uses an asymp-

totic maximum likelihood estimate of the inner product, then as the number of columns of R increases, the estimate of the inner product would be more accurate.

Thus, it is reasonable to use RPCV for Euclidean distances, and Li's method for inner products.

5.5.2 Experiments With Real Data

We now demonstrate RPCV on four datasets, the `arcene` dataset, `colon` dataset, `kos` dataset, and the `NIPS` dataset. More information about these datasets can be found in Appendix A. We select these datasets since they have different characteristics (sparse / dense, variance explained / not explained in few principal components).

We normalize each dataset such that every observation $\|\mathbf{x}_i\|_2^2 = 1$.

For each dataset, we consider the pairwise Euclidean distances of all observations $\{\mathbf{x}_i, \mathbf{x}_j\}, \quad \forall\ i \neq j$, and compute the estimates of the Euclidean distance with RPCV of the pairs $\{\mathbf{x}_i, \mathbf{x}_j\}$ which give the 20th, 30th, ..., 90th percentile of Euclidean distances.

We first do a quick sanity check in Figure 5.5. Here, we pick a pair in the 50th percentile for these datasets and show that for every different R_i, the bias quickly converges to zero.

Next, we look at the variance reduction for these pairs in Figure 5.6 with different types of random projection matrices R_i. We see that the variance re-

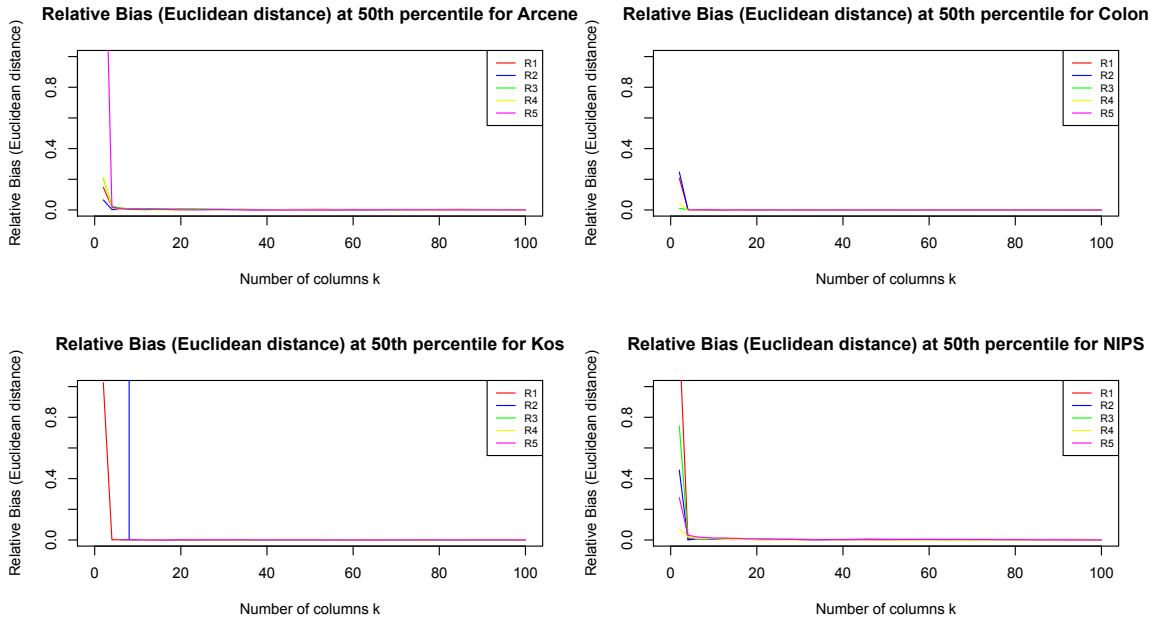

Figure 5.5: Plots Of Relative Bias In Euclidean Distance For Real Data

duction for the R_is are around the same range. Since the bias converges to zero, this implies that our control variates work. i.e., we do not get extremely biased estimates with lower variance.

We now look at what happens at different percentile pairs. Since the random projection matrices have a similar pattern in Figure 5.6, we will only take a look at varying pairs for the random projection matrix R_1.

Figure 5.7 thus shows the ratio ρ of variance reduction from the 10th percentile to the 90th percentile. Note that for dense datasets (`arcene`, `colon`), we can see a substantial percentage increase in variance reduction as the percentiles increase, but not as much for sparse datasets (`kos`, `NIPS`).

Finally, we take a look at the inner product estimates. Unfortunately, we do not get good variance reduction results, when we used Li's method as a

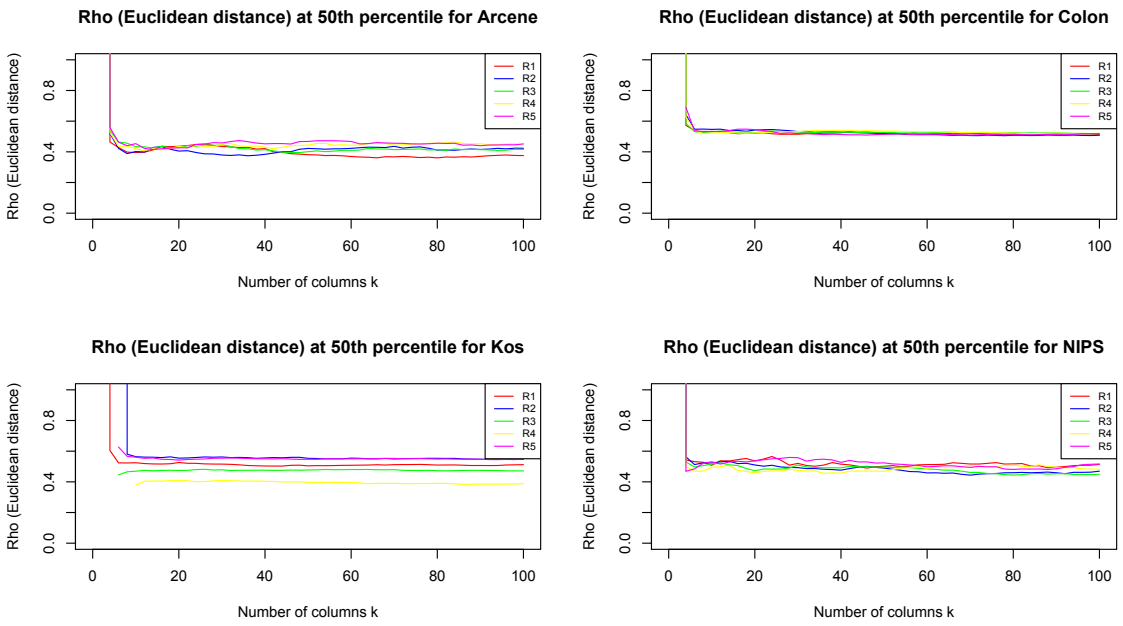

Figure 5.6: Plots Of ρ For Euclidean Distance (Varying R) For Real Data

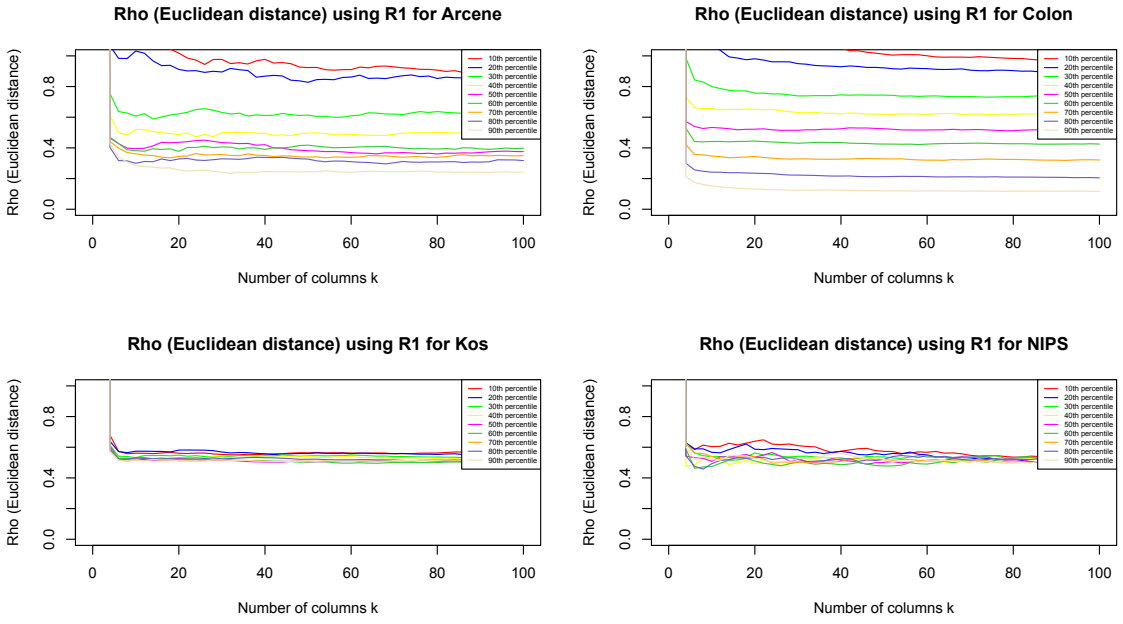

Figure 5.7: Plots Of ρ For Euclidean Distance (Varying Percentile) For Real Data

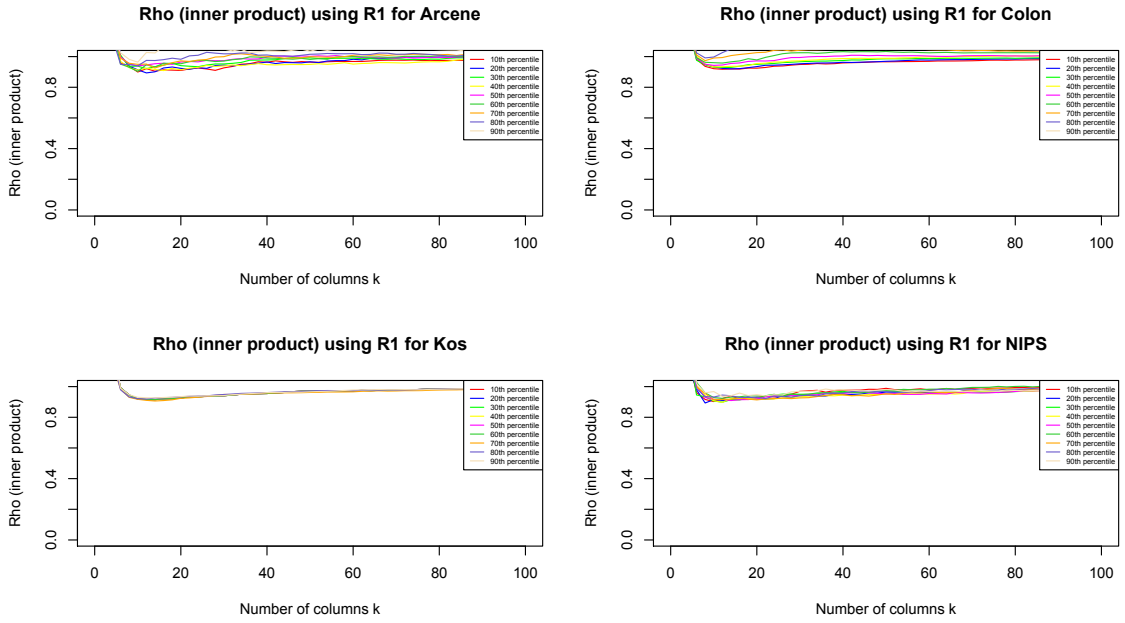

Figure 5.8: Plots Of ρ For Inner Product (Varying Percentile) For Real Data

baseline.

Figure 5.8 shows the plots of ρ with the random projection matrix R_1 for varying percentiles. The same pattern holds for different types of matrices R_2 to R_4, and is similar to what we saw in synthetic data. While there is some small variance reduction, the ratio ρ quickly converges to a value near 1.

This matches what we see in our synthetic data.

5.6 Conclusion

We have presented a new method RPCV which works well in conjunction with different random projection matrices to reduce the variance of the estimates of

the Euclidean distance and inner products on different types of vectors $\mathbf{x}_i$, $\mathbf{x}_j$. This allows for more accurate estimates of the Euclidean distance. As the Euclidean distance between two vectors increases, we expect greater variance reduction. In essence, we have shown that it is possible to juxtapose statistical variance reduction methods with random projections to give better results.

While RPCV gives a variance reduction for the estimates of the inner products, the ratio of variance reduction becomes minimal as the number of columns increases when compared to Li's method. This is not surprising since Li's method for estimating the inner products is an asymptotic maximum likelihood estimator, and is extremely accurate as the number of columns increases.

Although RPCV requires storing marginal norms and computing the covariance between two p dimensional vectors, the cost of doing so is negligible when compared to matrix multiplication. Furthermore, the computation of marginal norms is unnecessary when the data is already normalized.

In fact, RPCV can be seen as a method that nicely complements Li's method since both methods require storing marginal norms. RPCV substantially reduces the errors of the estimates of the Euclidean distance, while Li's method substantially reduces the errors of the estimates of the inner product.

We note that different applications may require a certain type of random projection matrix. Thus if we want to reduce the errors in our estimates, we cannot just switch to a different random projection matrix where the entries allow us to place sharper probability bounds on our errors. If we want data to be invariant under rotations, then a Normal random projection matrix would be best suited [16]. If we wanted to desparsify data, then a random projection

matrix with i.i.d. entries from $\{-\sqrt{p}, 0, \sqrt{p}\}$, p small might be preferred [1]. If we are focused on speed and quick information retrieval, then very sparse random projections [12] or random projection matrices formed by the SHRT [4] would be more preferable. RPCV allows us to reduce the error in all these estimates.

Finally, we look forward to extending this method of control variates to other applications of random projections, and further improving this method using extra marginal information.

Future work can be done where we look at random projection with multiple control variates.

5.7 Future Work

Multiple control variates can be seen as an extension of single control variates. Instead of having the expression:

$$\mathbb{E}[A + c(B - \mu_B)] = \mathbb{E}[A] \tag{5.54}$$

defined in Equation 5.3, we could consider the expression:

$$\mathbb{E}\left[A + \sum_{i=1}^{n} c_i(B_i - \mu_{B_i})\right] = \mathbb{E}[A] \tag{5.55}$$

In the case of random projections, we can write:

$$\mathbb{E}\left[f(v_{is}, v_{js}) + \sum_{t=1}^{n} c_{it} g(\mathbf{x}_i, \mathbf{e}_t) + c_{jt} g(\mathbf{x}_j, \mathbf{e}_t)\right] = \mathbb{E}[f(v_i, v_{,j})] \tag{5.56}$$

where we have:

$$\begin{aligned}
\mathbf{x}_i, \mathbf{x}_j &: \text{ pair of vectors we are considering} \\
f(v_{is}, v_{ij}) &: \text{ a realization of either Euclidean distance or inner product} \\
g(\mathbf{x}_\cdot, \mathbf{e}_t) &: \text{ the inner product of } \mathbf{x}_\cdot \text{ with the } t^{\text{th}} \text{ principal component} \\
c_{\cdot\cdot} &: \text{ optimal } c_{\cdot\cdot} \text{ parameter to be found}
\end{aligned}$$

These $c_{\cdot\cdot}$ can be found by numerical optimization methods.

In practice, $g(\mathbf{x}_\cdot, \mathbf{e}_t)$ could be any plausible function, not necessarily the inner product. As we are trying to see if we can improve from Li's baseline for estimation of inner products, we propose g to be the inner product function.

However, this is slated for future work.

CHAPTER 6

CONCLUSION

In this thesis, we have reviewed a brief history of random projections in the first three chapters.

We looked at random projections as a type of optimization algorithm based on variances, and realized that tweaks to the random projection structure on its own cannot bring us much further. At the very most,[17, 28] we might gain an improvement for one dataset, but then discover the algorithm works poorly on another dataset for that same tweak.

Performance (accuracy) guarantees aside, we did see that changes to the random projection structure can bring about memory and speed advantages.

For example, we would get a faster computation time if entries were stored as $0, 1$ entries rather than dense entries. Achlioptas and Li demonstrated this with binary random projections and very sparse random projections respectively.

Building onto this, we would also get a faster time computing V, where $V = XR$ if we considered recursive structures of random matrices. The Subsampled Randomized Hadamard Transform (SRHT) [4] is an example of that.

Thus it is possible to work around the No Free Lunch theorem. Suppose it took k columns for a random projection matrix with entries from $N(0, 1)$ to get a certain degree of accuracy. Further suppose that it would take $k + \alpha$ columns for a random projection matrix with a different structure to get the same degree of accuracy. It would be worthwhile if the time taken to generate the latter random

projection matrix is still faster than the former.

In chapter 3, we proposed BCD random projections which can be seen as random projections with "hyper-parameters". In this context, a random projection matrix with a certain structure would be generated based on properties of a dataset. This random projection matrix would guarantee a more accurate estimate of the quantities we want to compute. However, the pre-processing cost of actually finding such a random projection matrix may be prohibitive. Thus, it might not necessarily be useful to use BCD random projections for a slight gain in accuracy, if the trade-off is a much longer pre-processing time.

In Chapter 4, we discussed Qi's and Farhad's [20, 22] algorithms in reconstructing principal components using random projections. Building upon previous work in the random projection literature, we focused on reducing the storage space needed for the random projections R_i and the computational cost. We do not claim that our methods will always outperform Qi and Farhad, even though our empirical results with the MNIST dataset seems to suggest so.

Rather, we make the claim that our method further reduces the storage cost of random projections, and can even provide a speedup by taking advantage of the Hadamard matrix structure. These benefits may outweigh the disadvantage in accuracy on "bad" datasets.

Finally, we considered control variates in Chapter 5 in order to reduce the variance in the estimates of Euclidean distance and inner products. We see that control variates does give a substantial variance reduction in these estimates for four different types of datasets and different random projection matrix structure.

As computational time taken for control variates is of the same order as ordinary random projections, then control variates should be used in conjunction with random projections to get more accurate estimates.

We also saw that using control variates with principal components also provides variance reduction in datasets where the first principal component accounts for a reasonable portion of the variance, but doesn't provide variance reduction otherwise. This again highlights [17], but in this case it is easier to identify favourable datasets where the first few principal components does account for most of the variance.

Overall, we suggested modifications to random projection algorithms which try to account for the No Free Lunch theorem (BCD random projections, random projections with control variates) or work around the No Free Lunch theorem by trading off accuracy with an increase in speed / decrease in memory use (Semi Deterministic Random Projections). We hope that these modifications would be useful to people who work with random projections.

APPENDIX A

DATASETS

We use five data sets in this thesis to demonstrate empirical results. A brief description of these datasets can be found in this appendix.

A.1 Arcene Dataset

The Arcene dataset comes from Guyon et al [7], which contains mass-spectrometric data of cancer and non cancer patients.

There are $n = 900$ observations (patients) in this dataset, and $p = 10000$ features, with two different classes (cancer / non cancer).

While $p = 10000$, most of the variance in this dataset is explained by the first 500 eigenvectors. About 50% of the variance in the data is explained by the first 10 eigenvectors, with the first eigenvector explaining slightly more than 20% of the variance. Figure A.1 shows a scree plot of all $p = 10000$ eigenvectors, and a scree plot of the first 10 eigenvectors.

This dataset can be found online at `https://archive.ics.uci.edu/ml/datasets/Arcene` [14].

A.2 Colon Dataset

The Colon dataset comes from Alon et al [3], which contains gene expression levels of 40 tumour and 22 normal colon tissues for 6500 human genes obtained

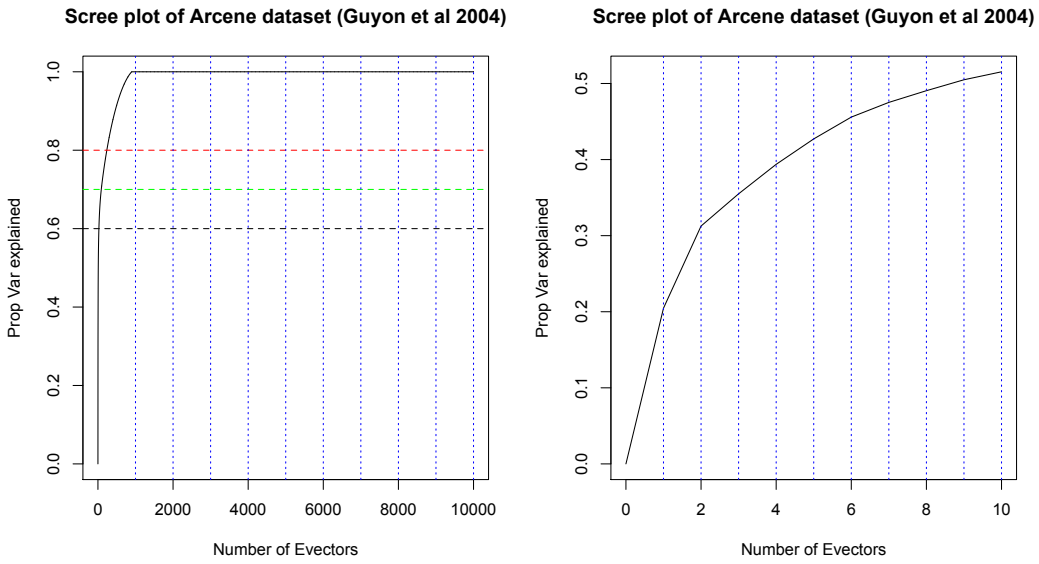

Figure A.1: Scree Plot Of Arcene Dataset

with an Affymetrix oligonucleotide array.

There are $n = 62$ observations (genes) in this dataset, and $p = 2000$ features, with two different classes (tumour / non tumour).

While $p = 2000$, most of the variance in this dataset is explained by a few eigenvectors. About 80% of the variance in the data is explained by the first 9 eigenvectors, with the first eigenvector explaining slightly more than 40% of the variance. Figure A.2 shows a scree plot of all $p = 2000$ eigenvectors, and a scree plot of the first 10 eigenvectors.

This dataset can be found online at `http://laurel.datsi.fi.upm.es/~vrobles/_export/xhtml/primero`. Alternatively, this dataset comes pre-loaded in the `R` package `plsgenomics`.

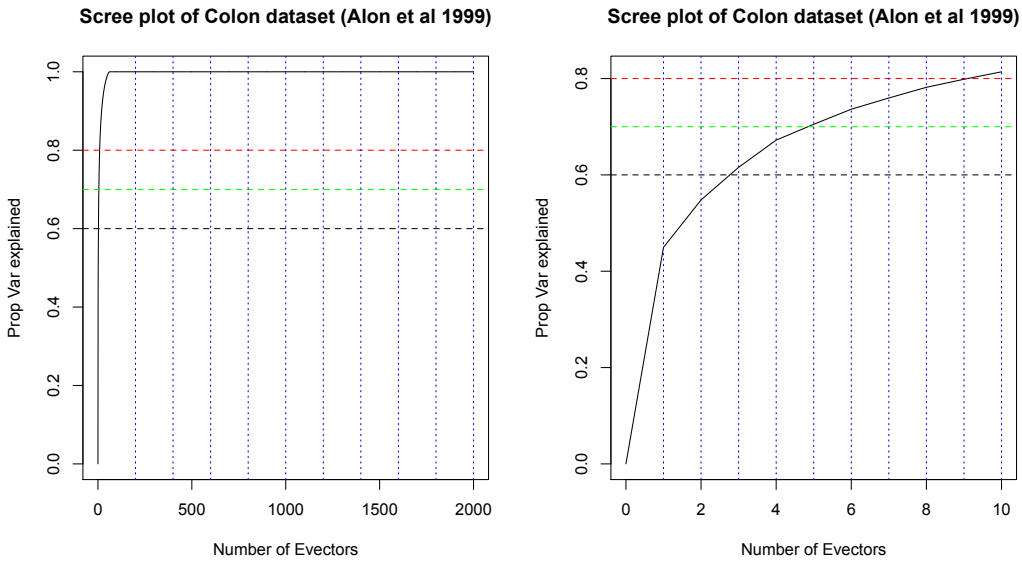

Figure A.2: Scree Plot Of Colon Dataset

A.3 Kos Blogposts Dataset

The Kos blogposts dataset is a set blog entries from the `dailykos.com` which contains 3430 documents, and 6906 words. Altogether, there are $n = 3430$ observations, and $p = 6906$ features. This is a relatively sparse dataset.

Most of the variance in this dataset is explained by about a third of the eigenvectors. About 80% of the variance in the data is explained by the first 1000 eigenvectors. Each individual eigenvector does not contribute much to the variance at all, with the first eigenvector contributing less than 10% of the variance, and the first ten eigenvectors contributing 20% of the variance. Figure A.3 shows a scree plot of all $p = 6906$ eigenvectors, and a scree plot of the first 10 eigenvectors.

This dataset can be found online at `https://archive.ics.uci.edu/`

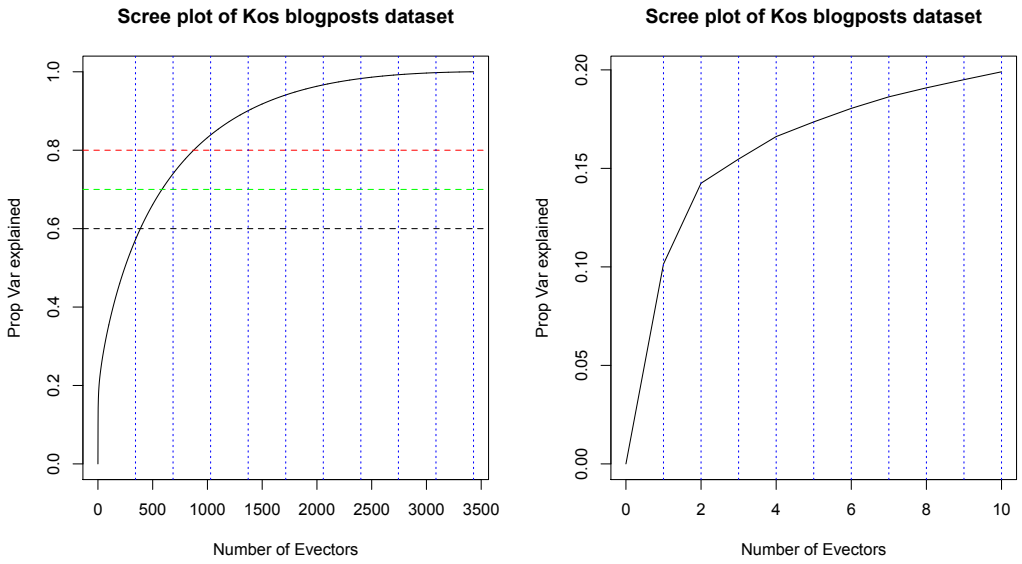

Figure A.3: Scree Plot Of Kos Blogposts Dataset

`ml/datasets/Bag+of+Words` [14].

A.4 MNIST Dataset

The MNIST dataset is hosted by Yann Le Cun on his website, which is a database of 70,000 handwritten digits from 0-9, with 60,000 training examples, and 10,000 test examples.

Altogether, there are $n = 70,000$ observations in this dataset, and $p = 784$ features (images are 28×28 pixels).

Most of the variance in this dataset is explained by the first 100 eigenvectors. The first ten eigenvectors contribute about 50% of the variance, with the first eigenvector contributing about 10% of the variance. Figure A.4 shows a scree

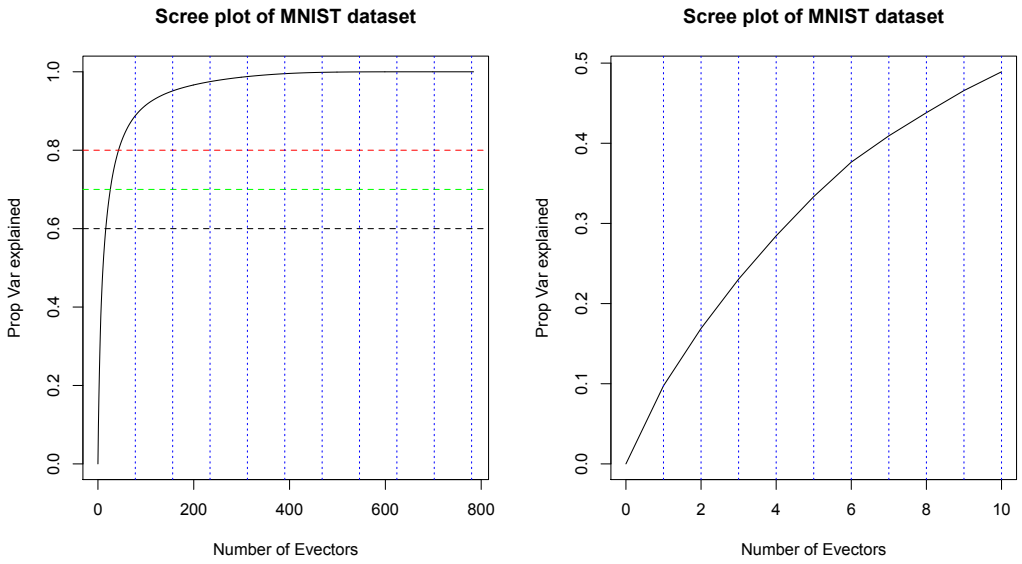

Figure A.4: Scree Plot Of MNIST Dataset

plot of all $p = 784$ eigenvectors, and a scree plot of the first 10 eigenvectors.

This dataset can be downloaded at `http://yann.lecun.com/exdb/mnist/`.

A.5 NIPS Conference Dataset

The NIPS conference dataset comes from Perrone et al [19], which contains contains the distribution of words in the full text of the NIPS conference papers published from 1987 to 2015. This is a relatively sparse dataset.

There are $n = 5812$ observations (conference papers) in this dataset, and $p = 11463$ words.

Most of the variance in this dataset is explained by slightly less than half

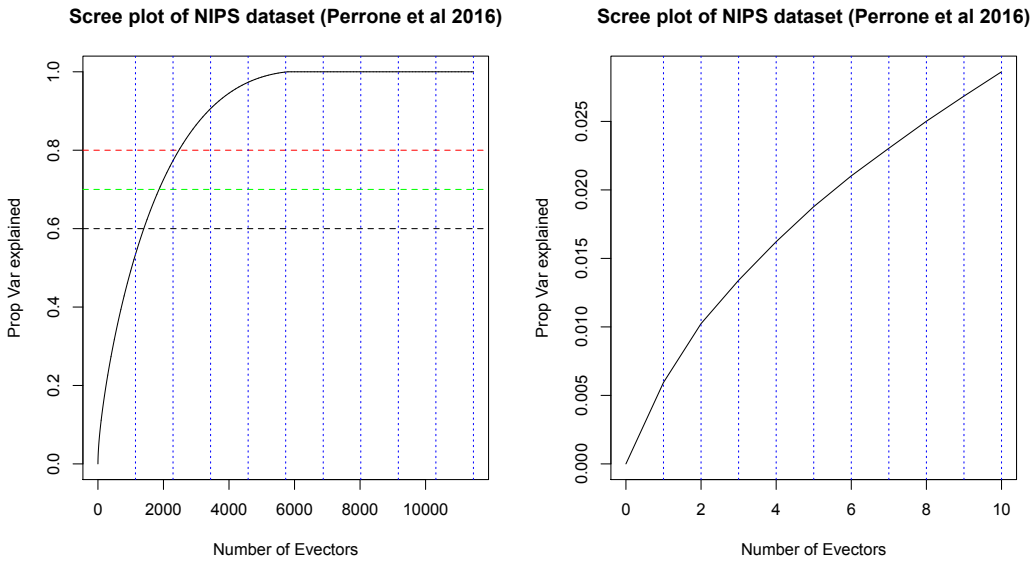

Figure A.5: Scree Plot Of NIPS Conference Dataset

of the eigenvectors. About 80% of the variance in the data is explained by the first 2000 eigenvectors. Each individual eigenvector does not contribute much to the variance at all, with the first eigenvector contributing less than 0.5% of the variance. Figure A.5 shows a scree plot of all $p = 11463$ eigenvectors, and a scree plot of the first 10 eigenvectors.

This dataset can be found online at `https://archive.ics.uci.edu/ml/datasets/NIPS+Conference+Papers+1987-2015` [14].

APPENDIX B

LEMMA TO COMPUTE MOMENTS

Most of the work in this thesis requires computing the second and fourth moments of certain expressions. We give a lemma that simplifies this computation.

Lemma B.0.1. Suppose we have a sequence of terms $\{t_i\}_{i=1}^p = \{a_i r_i\}_{i=1}^p$ for $\mathbf{a} = (a_1, a_2, \ldots, a_p)$, $\{s_i\}_{i=1}^p = \{b_i r_i\}_{i=1}^p$ for $\mathbf{b} = (b_1, b_2, \ldots, b_p)$ and r_i i.i.d. random variables with odd moments $\mathbb{E}[r_i^{2k+1}] = 0$ and finite even moments $\mathbb{E}[r_i^{2k}] = \mu_{2k}$. Then:

$$\mathbb{E}\left[\left(\sum_{i=1}^p t_i\right)^2\right] = \mu_2 \sum_{i=1}^p a_i^2 = \mu_2 \|\mathbf{a}\|_2^2 \tag{B.1}$$

$$\mathbb{E}\left[\left(\sum_{i=1}^p t_i\right)^4\right] = \mu_4 \sum_{i=1}^p a_i^4 + 6\mu_2^2 \sum_{u=1}^{p-1} \sum_{v=u+1}^p a_u^2 a_v^2 \tag{B.2}$$

$$\mathbb{E}\left[\left(\sum_{i=1}^p s_i\right)\left(\sum_{i=1}^p t_i\right)\right] = \mu_2 \sum_{i=1}^p a_i b_i = \mu_2 \langle \mathbf{a}, \mathbf{b} \rangle \tag{B.3}$$

$$\mathbb{E}\left[\left(\sum_{i=1}^p s_i\right)^2 \left(\sum_{i=1}^p t_i\right)^2\right] = \mu_2^2 \left(\sum_{i=1}^p a_i^2 b_i^2 + \sum_{i \neq j} a_i^2 b_j^2 + 4 \sum_{u=1}^{p-1} \sum_{v=u+1}^p a_u b_u a_v b_v\right) \tag{B.4}$$

Proof. We first note that any terms of the form: $\mathbb{E}[t_i t_j^n]$ are equal to zero for $i \neq j$, since we can write:

$$\mathbb{E}[t_i t_j^n] = \mathbb{E}[t_i]\mathbb{E}[t_j^n] = 0 \tag{B.5}$$

With this in mind, it is straightforward to note that:

$$\mathbb{E}\left[\left(\sum_{i=1}^p t_i\right)^2\right] = \mathbb{E}\left[\sum_{i=1}^p t_i^2 + 2 \sum_{u=1}^{p-1} \sum_{v=u+1}^p t_u t_v\right] \tag{B.6}$$

$$= \sum_{i=1}^p \mathbb{E}[t_i^2] + 2 \sum_{u=1}^{p-1} \sum_{v=u+1}^p \mathbb{E}[t_u]\mathbb{E}[t_v] \tag{B.7}$$

$$= \mu_2 \sum_{i=1}^p a_i^2 \tag{B.8}$$

and we have proved Equation B.1.

Next we write:

$$\mathbb{E}\left[\left(\sum_{i=1}^{p} t_i\right)^4\right] = \mathbb{E}\left[\left(\sum_{i=1}^{p} t_i^2 + 2\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} t_i t_j\right)^2\right] \tag{B.9}$$

$$= \mathbb{E}\left[\left(\sum_{i=1}^{p} t_i^2\right)^2 + 4\left(\sum_{i=1}^{p} t_i^2\right)\left(\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} t_i t_j\right) + 4\left(\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} t_i t_j\right)^2\right] \tag{B.10}$$

The first expression on the RHS of Equation B.10 simplifies to:

$$\mathbb{E}\left[\left(\sum_{i=1}^{p} t_i^2\right)^2\right] = \mathbb{E}\left[\sum_{i=1}^{p} t_i^4 + 2\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} t_u^2 t_v^2\right] \tag{B.11}$$

$$= \sum_{i=1}^{p} \mathbb{E}[t_i^4] + 2\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} \mathbb{E}[t_u^2]\mathbb{E}[t_v^2] \tag{B.12}$$

$$= \mu_4 \sum_{i=1}^{p} a_i^4 + 2\mu_2^2 \sum_{u=1}^{p-1}\sum_{v=u+1}^{p} a_u^2 a_v^2 \tag{B.13}$$

For the second expression on the RHS of Equation B.10, note that in the second bracket, we have $t_i \neq t_j$ since $i \neq j$, and thus multiplying the terms in the first and second bracket together, we will either have three distinct terms in the (sum of the) expression, say $t_i t_j t_k$ with $i \neq j,\ i \neq k,\ j \neq k$, or some $t_i^2 t_j$ with $i \neq j$. The expectation goes to zero in either case.

The third expression on the RHS of Equation B.10 simplifies to:

$$4\mathbb{E}\left[\left(\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} t_i t_j\right)^2\right] = 4\mathbb{E}\left[\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} t_u^2 t_v^2 + \sum_{u,v,w} t_u t_v t_w\right] \tag{B.14}$$

$$= 4\left(\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} \mathbb{E}[t_u^2]\mathbb{E}[t_v^2] + \sum_{u,v,w} \mathbb{E}[t_u]\mathbb{E}[t_v]\mathbb{E}[t_w]\right) \tag{B.15}$$

$$= 4\mu_2^2 \sum_{u=1}^{p-1}\sum_{v=u+1}^{p} a_u^2 a_v^2 \tag{B.16}$$

The second term $\sum_{u,v,w} \mathbb{E}[t_u]\mathbb{E}[t_v]\mathbb{E}[t_w]$ will either have some $t_u t_v t_w$ where $u \neq v,\ u \neq w,\ v \neq w$, or some $\mathbb{E}[t_i^2]\mathbb{E}[t_j]$ where $i \neq j$. In either case, the expectation of this term goes to zero as well.

Putting our results together, we have:

$$\mathbb{E}\left[\left(\sum_{i=1}^{p} t_i\right)^4\right] = \mu_4 \sum_{i=1}^{p} a_i^4 + 6\mu_2^2 \sum_{u=1}^{p-1} \sum_{v=u+1}^{p} a_u^2 a_v^2 \tag{B.17}$$

and we have proved Equation B.2.

Third, we note that:

$$\left(\sum_{i=1}^{p} s_i\right)\left(\sum_{i=1}^{p} t_i\right) = \mathbf{s}\mathbf{t}^T \tag{B.18}$$

is simply the outer product of these two vectors, and can be written as:

$$\left(\sum_{i=1}^{p} s_i\right)\left(\sum_{i=1}^{p} t_i\right) = \sum_{i=1}^{p} s_i t_i + \sum_{i \neq j} s_i t_j \tag{B.19}$$

Therefore:

$$\mathbb{E}\left[\left(\sum_{i=1}^{p} s_i\right)\left(\sum_{i=1}^{p} t_i\right)\right] = \sum_{i=1}^{p} \mathbb{E}[s_i]\mathbb{E}[t_i] + \sum_{i \neq j} \mathbb{E}[s_i]\mathbb{E}[t_j] \tag{B.20}$$

$$= \sum_{i=1}^{p} a_i b_i \mathbb{E}[r_i^2] + 2 \sum_{i \neq j} a_i b_j \mathbb{E}[r_i]\mathbb{E}[r_j] \tag{B.21}$$

$$= \mu_2 \sum_{i=1}^{p} a_i b_i \tag{B.22}$$

and we have proved Equation B.3.

Lastly, we consider:

$$\left(\sum_{i=1}^{p} s_i\right)^2 \left(\sum_{i=1}^{p} t_i\right)^2 = \left(\sum_{i=1}^{p} s_i^2 + 2 \sum_{u=1}^{p-1} \sum_{v=u+1}^{p} s_u s_v\right)\left(\sum_{i=1}^{p} t_i^2 + 2 \sum_{u=1}^{p-1} \sum_{v=u+1}^{p} t_u t_v\right) \tag{B.23}$$

which can be split up into a sum of four terms.

The first term gives:

$$\left(\sum_{i=1}^{p} s_i^2\right)\left(\sum_{i=1}^{p} t_i^2\right) = \sum_{i=1}^{p} s_i^2 t_i^2 + \sum_{i \neq j} s_i^2 t_j^2 \tag{B.24}$$

and taking the expectation yields:

$$\mathbb{E}\left[\sum_{i=1}^{p} s_i^2 t_i^2 + \sum_{i\neq j} s_i^2 t_j^2\right] = \sum_{i=1}^{p} \mathbb{E}[s_i^2]\mathbb{E}[t_i^2] + \sum_{i\neq j} \mathbb{E}[s_i^2]\mathbb{E}[t_j^2] \tag{B.25}$$

$$= \mu_2^2 \sum_{i=1}^{p} a_i^2 b_i^2 + \mu_2^2 \sum_{i\neq j} a_i^2 b_j^2 \tag{B.26}$$

The second and third terms give

$$2\left(\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} s_u s_v\right)\left(\sum_{i=1}^{p} t_i^2\right) \tag{B.27}$$

and:

$$2\left(\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} t_u t_v\right)\left(\sum_{i=1}^{p} s_i^2\right) \tag{B.28}$$

respectively. If we think about the indices of the cross terms, we either get some form of $t_i^2 s_j s_k$, or $t_i^2 s_i s_k$, which in both cases has a single index on its own. Thus, the expectation of either $\mathbb{E}[t_i^2 s_j s_k]$ or $\mathbb{E}[t_i^2 s_i s_k] = 0$.

Finally, the last term gives:

$$4\left(\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} s_u s_v\right)\left(\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} t_u t_v\right) \tag{B.29}$$

If we think about the indices of the cross terms, then the only case where we get a non-zero expectation is when we have cross terms of the form $s_u s_v t_u t_v$.

Therefore, we have:

$$\mathbb{E}\left[4\left(\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} s_u s_v\right)\left(\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} t_u t_v\right)\right] = 4\mu_2^2 \sum_{u=1}^{p-1}\sum_{v=u+1}^{p} a_u b_u a_v t_v \tag{B.30}$$

Putting everything together, we have:

$$\mathbb{E}\left[\left(\sum_{i=1}^{p} s_i\right)^2\left(\sum_{i=1}^{p} t_i\right)^2\right] = \mu_2^2\left(\sum_{i=1}^{p} a_i^2 b_i^2 + \sum_{i\neq j} a_i^2 b_j^2 + 4\sum_{u=1}^{p-1}\sum_{v=u+1}^{p} a_u b_u a_v b_v\right) \tag{B.31}$$

and we have proved Equation B.4. □

APPENDIX C

VARIANCE PROOFS FOR BCD RANDOM PROJECTIONS

We give the proofs for Theorem 3.2.1, Theorem 3.2.3, and Theorem 3.2.4.

Theorem 3.2.1. For $R(B,1)$ with k columns, we have that

$$\mathrm{Var}[\|\mathbf{v}\|^2] = 4B\left(\sum_{i=0}^{B-1}\left\{\sum_{a,b\geq\frac{ip}{B}+1}^{\frac{p(i+1)}{\beta}} x_{1a}^2 x_{1b}^2\right\}\right) \tag{C.1}$$

Proof. We consider the case where $\beta = 2$, and then generalize this to the case where $\beta = B$. We note that $2|k$, and assume $a|p$ for easier writing of indices, as the mechanics of the proof is the same otherwise.

Our set up is given by:

$$\begin{pmatrix} v_{11} & v_{12} & \dots & v_{1k} \\ v_{21} & v_{22} & \dots & v_{2k} \\ \dots & \dots & \ddots & \dots \\ v_{n1} & v_{n2} & \dots & v_{nk} \end{pmatrix} = \sqrt{2}\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \dots & \dots & \ddots & \dots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix}\begin{pmatrix} r_{11} & 0 & \dots & 0 \\ r_{21} & 0 & \dots & 0 \\ \dots & \dots & \ddots & \dots \\ 0 & r_{p1} & \dots & r_{p\frac{k}{2}} \end{pmatrix} \tag{C.2}$$

We note that since r_{ij}s are independent, it suffices to consider v_{11} and v_{12}.

To calculate $\mathrm{Var}[v_{11}^2 + v_{12}^2]$, we need to find $\mathbb{E}[(v_{11}^2 + v_{12}^2)^2] = \mathbb{E}[v_{11}^4 + 2v_{11}^2 v_{12}^2 + v_{12}^4]$.

From Lemma B.0.1, we have:

$$\mathbb{E}[v_{11}^4] = 4\left(\sum_{s=1}^{p/2} x_{1s}^4 + 6\sum_{a,b\geq 1}^{p/2} x_{1a}^2 x_{1b}^2\right) \tag{C.3}$$

$$\mathbb{E}[v_{12}^4] = 4\left(\sum_{s=\frac{p}{2}+1}^{p} x_{1s}^4 + 6\sum_{a,b\geq\frac{p}{2}+1}^{p} x_{1a}^2 x_{1b}^2\right) \tag{C.4}$$

$$2\mathbb{E}[v_{11}^2 v_{12}^2] = 8\left(\sum_{k=1}^{p/2} x_{1k}^2\right)\left(\sum_{k=\frac{p}{2}+1}^{p} x_{1k}^2\right) \tag{C.5}$$

and:

$$\left(\mathbb{E}[v_{11}^2 + v_{12}^2]\right)^2 = 4\left(\sum_{k=1}^{p} x_{1k}^4 + 2\sum_{a,b}^{p} x_{1a}^2 x_{1b}^2\right) \tag{C.6}$$

Thus:

$$\text{Var}[(v_{11}^2 + v_{12}^2)] = 4\left(\sum_{s=1}^{p} x_{1s}^4 + 6\left(\sum_{a,b\geq 1}^{p/2} x_{1a}^2 x_{1b}^2 + \sum_{a,b\geq \frac{p}{2}+1}^{p} x_{1a}^2 x_{1b}^2\right)\right. \tag{C.7}$$

$$\left. +2\left(\sum_{k=1}^{p/2} x_{1k}^2\right)\left(\sum_{k=\frac{p}{2}+1}^{p} x_{1k}^2\right) - \sum_{k=1}^{p} x_{1k}^4 - 2\sum_{a,b}^{p} x_{1a}^2 x_{1b}^2\right) \tag{C.8}$$

$$= 16\left(\sum_{a,b\geq 1}^{p/2} x_{1a}^2 x_{1b}^2 + \sum_{a,b\geq \frac{p}{2}+1}^{p} x_{1a}^2 x_{1b}^2\right) \tag{C.9}$$

since $\sum_{a,b}^{p} x_{1a}^2 x_{1b}^2 = \sum_{a,b\geq 1}^{\frac{p}{2}} x_{1a}^2 x_{1b}^2 + \sum_{a,b\geq \frac{p}{2}+1}^{p} x_{1a}^2 x_{1b}^2 + \left(\sum_{k=1}^{p/2} x_{1k}^2\right)\left(\sum_{k=\frac{p}{2}+1}^{p} x_{1k}^2\right)$.

We then get:

$$\text{Var}[(v_{11}^2 + \ldots + v_{1k}^2)] = 8\left(\sum_{a,b\geq 1}^{p/2} x_{1a}^2 x_{1b}^2 + \sum_{a,b\geq \frac{p}{2}+1}^{p} x_{1a}^2 x_{1b}^2\right) \tag{C.10}$$

It is easy to see (following the same steps from above), and noting that:

$$\sum_{a,b}^{p} x_{1a}^2 x_{1b}^2 = \sum_{i=0}^{\beta-1}\left\{\sum_{a,b\geq \frac{ip}{\beta}+1}^{\frac{p(i+1)}{\beta}} x_{1a}^2 x_{1b}^2\right\} + \sum_{i,j\geq 0}^{\beta-1}\left\{\left(\sum_{k=\frac{ip}{\beta}+1}^{\frac{(i+1)p}{\beta}} x_{1k}^2\right)\left(\sum_{k=\frac{jp}{\beta}+1}^{\frac{(j+1)p}{\beta}} x_{1k}^2\right)\right\} \tag{C.11}$$

we must have that for the general case with $\beta = B$, we have that:

$$\text{Var}[(v_{11}^2 + \ldots + v_{1k}^2)] = 4B\left(\sum_{i=0}^{\beta-1}\left\{\sum_{a,b\geq \frac{ip}{\beta}+1}^{\frac{p(i+1)}{\beta}} x_{1a}^2 x_{1b}^2\right\}\right) \tag{C.12}$$

□

Theorem 3.2.3. For $R(1, C)$ with k columns, we have that for a group of $C = 2^m$ columns, it is possible to remove at least $\frac{C-1}{C}\binom{p}{2}$ cross terms by placing the

negative signs carefully. We have

$$\mathrm{Var}[v_{11}^2 + \ldots + v_{1C}^2] = 4\sum_{s,t} x_{1s}^2 x_{1t}^2 - \frac{8}{C}\sum_{i,j}\left(\sum_{s,t} x_{1s}x_{1t}H_{si}H_{sj}H_{ti}H_{tj}\right) \tag{C.13}$$

Proof. Let us look at the estimate of the norm of $\|\mathbf{x}\|_2^2$. Without loss of generality, let us consider v_{11}, v_{12}, v_{13}, v_{14} for the cases $C = 2$ and $C = 4$, before proceeding on for general C.

We denote H_{ij} to be the sign in the i, j^{th} term of the Hadamard matrix.

Writing out the terms, we have:

$$v_{11} = \sum_{s=1}^{p} x_{1s}r_s H_{s1} \tag{C.14}$$

$$v_{12} = \sum_{s=1}^{p} x_{1s}r_s H_{s2} \tag{C.15}$$

$$v_{13} = \sum_{s=1}^{p} x_{1s}r_s H_{s3} \tag{C.16}$$

$$v_{14} = \sum_{s=1}^{p} x_{1s}r_s H_{s4} \tag{C.17}$$

with:

$$v_{11}^2 = \left(\sum_{s=1}^{p} x_{1s}^2 + 2\sum_{s,t} x_{1s}x_{1t}r_s r_t H_{s1}H_{t1}\right) \tag{C.18}$$

$$v_{12}^2 = \left(\sum_{s=1}^{p} x_{1s}^2 + 2\sum_{s,t} x_{1s}x_{1t}r_s r_t H_{s2}H_{t2}\right) \tag{C.19}$$

$$v_{13}^2 = \left(\sum_{s=1}^{p} x_{1s}^2 + 2\sum_{s,t} x_{1s}x_{1t}r_s r_t H_{s3}H_{t3}\right) \tag{C.20}$$

$$v_{14}^2 = \left(\sum_{s=1}^{p} x_{1s}^2 + 2\sum_{s,t} x_{1s}x_{1t}r_s r_t H_{s4}H_{t4}\right) \tag{C.21}$$

This leads to:

$$\mathbb{E}[v_{11}^2 + v_{12}^2] = 2\left(\sum_{s=1}^{p} x_{1s}^2\right) \tag{C.22}$$

$$\mathbb{E}[v_{11}^2 + v_{12}^2 + v_{13}^2 + v_{14}^2] = 4\left(\sum_{s=1}^{p} x_{1s}^2\right) \tag{C.23}$$

and in general:

$$\mathbb{E}[v_{11}^2 + \ldots + v_{1C}^2] = C\left(\sum_{s=1}^{p} x_{1s}^2\right) \tag{C.24}$$

$$\left(\mathbb{E}[v_{11}^2 + \ldots + v_{1C}^2]\right)^2 = C^2\left(\sum_{s=1}^{p} x_{1s}^4 + 2\sum_{s,t} x_{1s}^2 x_{1t}^2\right) \tag{C.25}$$

On the other hand, we have:

$$\mathbb{E}[v_{11}^4] = \mathbb{E}[v_{12}^4] = \mathbb{E}[v_{13}^4] = \mathbb{E}[v_{14}^4] = \left(\sum_{s=1}^{p} x_{1s}^4 + 6\sum_{s,t} x_{1s}^2 x_{1t}^2\right) \tag{C.26}$$

with any

$$\mathbb{E}[2v_{1i}^2 v_{1j}^2] = 2\left(\sum_{s=1}^{p} x_{1s}^2 + 2\sum_{s,t} x_{1s}^2 x_{1t}^2 - 4\sum_{s,t} x_{1s}x_{1t}H_{si}H_{sj}H_{ti}H_{tj}\right) \tag{C.27}$$

Now, noting that for any $(t_1 + t_C)^2$ we will have $\binom{C}{2} = \frac{C(C-1)}{2}$ cross terms, then we have

$$\mathbb{E}[(v_{11}^2 + \ldots + v_{1C}^2)^2] - (\mathbb{E}[v_{11}^2 + \ldots + v_{1C}^2])^2 \tag{C.28}$$

$$= C\left(\sum_{s=1}^{p} x_{1s}^4 + 6\sum_{s,t} x_{1s}^2 x_{1t}^2\right) + 2\binom{C}{2}\left(\sum_{s=1}^{p} x_{1s}^4 + 2\sum_{s,t} x_{1s}^2 x_{1t}^2\right) \tag{C.29}$$

$$- 8\sum_{i,j}\left(\sum_{s,t} x_{1s}x_{1t}H_{si}H_{sj}H_{ti}H_{tj}\right) - C^2\left(\sum_{s=1}^{p} x_{1s}^4 + 2\sum_{s,t} x_{1s}^2 x_{1t}^2\right) \tag{C.30}$$

$$= C\left(\sum_{s=1}^{p} x_{1s}^4 + 2\sum_{s,t} x_{1s}^2 x_{1t}^2 + 4\sum_{s,t} x_{1s}^2 x_{1t}^2\right) + C(C-1)\left(\sum_{s=1}^{p} x_{1s}^4 + 2\sum_{s,t} x_{1s}^2 x_{1t}^2\right) \tag{C.31}$$

$$- 8\sum_{i,j}\left(\sum_{s,t} x_{1s}x_{1t}H_{si}H_{sj}H_{ti}H_{tj}\right) - C^2\left(\sum_{s=1}^{p} x_{1s}^4 + 2\sum_{s,t} x_{1s}^2 x_{1t}^2\right) \tag{C.32}$$

$$= 4C\sum_{s,t} x_{1s}^2 x_{1t}^2 - 8\sum_{i,j}\left(\sum_{s,t} x_{1s}x_{1t}H_{si}H_{sj}H_{ti}H_{tj}\right) \tag{C.33}$$

Then, we must have:

$$\text{Var}[v_{11}^2 + \ldots + v_{1C}^2] = 4C \sum_{s,t} x_{1s}^2 x_{1t}^2 - 8 \sum_{i,j} \left(\sum_{s,t} x_{1s} x_{1t} H_{si} H_{sj} H_{ti} H_{tj} \right) \tag{C.34}$$

□

Theorem 3.2.4. For $R(\beta, \rho)$ where $\beta, \rho \geq 2$, we have that:

$$\text{Var}\left[\|\mathbf{v}\|^2 \right] = 4C \sum_{b=1}^{B} \left(\sum_{s,t \geq B(b-1)}^{b\frac{p}{B}} x_{1s}^2 x_{1t}^2 \right) - 8 \sum_{b=1}^{B} \left(\sum_{i,j \geq B(b-1)}^{b\frac{p}{B}} \left(\sum_{s,t \geq B(b-1)}^{b\frac{p}{B}} x_{1s} x_{1t} H_{si} H_{sj} H_{ti} H_{tj} \right) \right) \tag{C.35}$$

Proof. This is a consequence of applying both Theorem 3.2.3 and Theorem 3.2.1 in conjunction with each other. □

BIBLIOGRAPHY

[1] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, June 2003.

[2] Nir Ailon and Bernard Chazelle. The fast johnson-lindenstrauss transform and approximate nearest neighbors. *SIAM J. Comput.*, 39(1):302–322, May 2009.

[3] U. Alon, N. Barkai, D.A. Notterman, K. Gish, S. Ybarra, D. Mack, and A.J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–6750, June 1999.

[4] Christos Boutsidis and Alex Gittens. Improved matrix algorithms via the subsampled randomized hadamard transform. *CoRR*, abs/1204.0062, 2012.

[5] Sanjoy Dasgupta. Experiments with random projection. *CoRR*, abs/1301.3849, 2013.

[6] J.E. Fowler. Compressive-Projection Principal Component Analysis. *Image Processing, IEEE Transactions on*, 18(10):2230–2242, Oct 2009.

[7] Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 545–552. MIT Press, 2005.

[8] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

[9] William Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, volume 26 of *Contemporary Mathematics*, pages 189–206. American Mathematical Society, 1984.

[10] Ping Li and Kenneth W. Church. A Sketch Algorithm for Estimating Two-Way and Multi-Way Associations. *Comput. Linguist.*, 33(3):305–354, September 2007.

[11] Ping Li, Trevor Hastie, and Kenneth Ward Church. Improving random projections using marginal information. In Gbor Lugosi and Hans-Ulrich Simon, editors, *COLT*, volume 4005 of *Lecture Notes in Computer Science*, pages 635–649. Springer, 2006.

[12] Ping Li, Trevor J. Hastie, and Kenneth W. Church. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 287–296, New York, NY, USA, 2006. ACM.

[13] Ping Li, Michael Mitzenmacher, and Anshumali Shrivastava. Coding for random projections. *CoRR*, abs/1308.2218, 2013.

[14] M. Lichman. UCI machine learning repository, 2013.

[15] Miles Lopes, Laurent Jacob, and Martin J Wainwright. A more powerful two-sample test in high dimensions using random projection. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1206–1214. Curran Associates, Inc., 2011.

[16] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Academic Press, 1979.

[17] G. D. Montañez. The Famine of Forte: Few Search Problems Greatly Favor Your Algorithm. *ArXiv e-prints*, September 2016.

[18] Saurabh Paul, Christos Boutsidis, Malik Magdon-Ismail, and Petros Drineas. Random projections for support vector machines. *CoRR*, abs/1211.6085, 2012.

[19] V. Perrone, P. A. Jenkins, D. Spano, and Y. W. Teh. Poisson random fields for dynamic feature models. ArXiv e-prints: 1611.07460, 2016.

[20] Farhad Pourkamali-Anaraki and Shannon M. Hughes. Memory and computation efficient pca via very sparse random projections. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1341–II–1349. JMLR.org, 2014.

[21] Terry Pratchett. *Hogfather*, volume 20 of *Discworld Series*. Victor Gollancz, 1996. featuring DEATH who TALKS LIKE THIS.

[22] Hanchao Qi and Shannon M. Hughes. Invariance of principal components under low-dimensional random projection of the data. In *19th IEEE International Conference on Image Processing, ICIP 2012, Lake Buena Vista, Orlando, FL, USA, September 30 - October 3, 2012*, pages 937–940, 2012.

[23] Sheldon M. Ross. *Simulation, Fourth Edition*. Academic Press, Inc., Orlando, FL, USA, 2006.

[24] Radhendushka Srivastava, Ping Li, and David Ruppert. Raptt: An exact two-sample test in high dimensions using random projections. *Journal of Computational and Graphical Statistics*, 25(3):954–970, 2016.

[25] Santosh Srinivas Vempala. *The random projection method*, volume 65 of *DIMACS series in discrete mathematics and theoretical computer science*. Providence, R.I. American Mathematical Society, 2004. Appendice p.101-105.

[26] Stanley L. Warner. Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. *Journal of the American Statistical Association*, 60(309):63+, March 1965.

[27] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1113–1120, New York, NY, USA, 2009. ACM.

[28] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *Trans. Evol. Comp*, 1(1):67–82, April 1997.